Embodied Understanding in Computer Programming
– A semiotic analysis of metaphors used in programming

Andreas Larsson

Handledare: Anders Jidesjö, Konrad Schönborn
Examinator: Emilia Fägerstam
Title: Embodied Understanding in Computer Programming – A semiotic analysis of metaphors used in programming
Författare: Andreas Larsson

Abstract/Sammanfattning
Language can be seen as a bridge between the way we think and our actions. From a social semiotic, language becomes a resource with which collective knowledge can be distributed. The way we talk about an artefact is a reflection of our conceptual understanding thereof. Often, we use conceptual metaphors to communicate abstract concepts in abstract reasoning domains such as computer science. Conceptual metaphors are formed in relation to primary metaphors and are grounded in sensorimotor experience. Primary metaphors can be used to analyse how we embody our conceptual understanding of specific concepts. The aims of this study are threefold: Firstly, to explore how conceptual metaphors are used by students and teachers while engaging in conversations about computer programming; secondly, to analyse the emergent conceptual metaphors to identify how teachers and students understand abstract aspects about computer programming and thirdly, to identify the sensorimotor experiences that contribute to shaping the conceptual metaphors used by the students and the teachers. Two teachers and three students from two different upper-secondary schools engaged in informal conversations regarding aspects of computer programming. The conversations were audio-recorded, and excerpts were transcribed verbatim and translated into English. The excerpts were analysed by adopting a methodological framework based on Grady's theory of primary metaphor. Results suggest that teachers and students have a multi-faceted conceptual understanding of computer science that involves concepts such as spatial relations, similarities between objects and computer code. Other emergent aspects concerned interpersonal relationships, in relation to customer service and social skills. The findings imply that expanding the number of available conceptual metaphors could lead to a more diverse set of didactic tools in computer science education, thus increasing overall conceptual understanding.


Nykkelord
Embodied Cognition, Computer Programming, Computer Science, Conceptual understanding, Conceptual metaphors, Social Semiotics, Qualitative analysis
Introduction and Aims of the Study .................................................................3

Understanding Computer Programming ..........................................................6

Methodological Framework – Using Social Semiotics for studying conceptual understanding in computer programming .................................................................10

Conceptual understanding manifested in language .........................................11

Embodied Cognition as a tool for analysing conceptual understanding ...........11

Grounding of primary and conceptual metaphors in cognition ......................12

The embodying of a metaphor in cognition ..................................................15

Analysing the embodying of a conceptual metaphor in cognition ..................18

Collecting information on how teachers and students understand aspects of computer programming .................................................................20

Results and Discussion ..................................................................................22

Emerging Metaphors Regarding Computer Programming and Activities ..........22

Emerging Metaphors Regarding Software Development ..................................25

Emerging Metaphors Regarding Algorithms ..................................................27

Emerging Metaphors Regarding Loops ............................................................30

Emerging Metaphors Regarding Different Types of Students .........................33

Reflecting on the emergent programming metaphors as a collective ..............37

Conclusions and Implications .......................................................................40

Acknowledgements .........................................................................................42

References ........................................................................................................43
Introduction and Aims of the Study

In everything we do there tends to be at least one basic guiding principle that provides us with an understanding of how we react to the constant stimuli that continuously flood our nervous system. Some of these principles are based on pure experience – try catching a ball by studying Newton’s laws of motions instead of playing catch. Other concepts, such as a visit to the dentist can be understood by using our previous experiences of an appointment at the doctor’s office. All we need to do is exchange the doctor with the dentist and in that sense, we know how to behave in, and what to expect from the specific situation (Lakoff & Johnson, 1980). Without this type of basic understanding of different concepts and situations as those above, we would have a difficult time not only interacting with, but also learning from the world around us. Our knowledge about concepts, although always present are nothing that we implicitly reflect upon on a daily basis. Instead, we are constantly organizing the information we perceive from the inside or the outside of our bodies, ultimately trying to organise and create meaning of everything that we perceive.

Concepts should not be seen as mere reflections of what we perceive as reality. Instead, concepts and our understanding thereof, are formed by our sensorimotor experiences and are embodied within our brains (Lakoff & Johnson, 1999). According to George Lakoff and Mark Johnson (1980), our conceptual understanding, is based on small linguistic units termed primary metaphors. For example, primary metaphors make it possible to use our experience of movement to understand the expression “we are running out of time” (Johnson, 2007). Furthermore, our experiences of movement to reason about concepts such as life in terms of a journey and death as life’s ultimate stop sign. Our primary metaphors – the rudimentary components of human reasoning – are constantly, most of the time unconsciously, being combined into more complex conceptual metaphors. This is what makes it possible for us to understand and reason about abstract subjects such as love, economics, politics, religious belief and computer science (Lakoff & Johnson, 1980; 1999).

The cognitive mechanism behind our conceptual understanding is the shaping of conceptual metaphors. This is why the study of metaphors is an integral part of understanding our embodied understanding of concepts (Lakoff & Turner, 1989; Lakoff & Johnson, 1999). Lakoff and Turner (1989) state that, “To study metaphor is to be confronted with hidden aspects of one’s mind and one’s own culture” (p. 214). The conceptual metaphor holds information about
how we experience the physical world and how we relate to it on a personal level, and by deconstructing conceptual metaphors, we are able to unveil underlying primary metaphors (Lakoff & Johnson, 1999).

Understanding and communicating concepts serves as a big part of understanding computer programming and getting an education in general. Classrooms are filled with literature and artefacts containing information and opportunities to practice the skills that students need to embody. This is, as stated above, a matter of previous sensorimotor experiences and how they relate to other concepts. By studying conceptual metaphors used in an educational environment it is possible, not only to untangle how students and teachers understand concepts, but also how conceptual understanding is grounded in implicit sensorimotor experiences.

The specific aims of this study are to:

i. Explore how conceptual metaphors are used by students and teachers while engaging in conversations about computer programming

ii. Use the emergent conceptual metaphors to identify how teachers and students understand abstract aspects of computer programming

iii. Identify the sensorimotor experiences that may contribute to shaping the conceptual metaphors used by the students and the teachers

Social semiotics and embodied cognition (Lakoff & Johnson, 1980, 1999; Danesi, 2007; Chandler, 2007) serve as the theoretical lens to respond to these aims. This framework is used to explore how five different individuals understand different concepts related to computer science and computer programming. The empirical data consists of transcripts of five different conversations concerning general aspects of computer science and computer programming. The participants are two teachers and three students, all working or studying in the third year of Teknikprogrammet. Since the aims of this study are exploratory, the conversations are held in an open fashion, allowing the respondents to manifest their own interpersonal relationship with computer science and computer programming without focusing on a particular situation, task or assignment.
The study is reported as follows. The first section provides a review of four metaphorical aspects of computer programming from recent literature on computer science education. The section closes by presenting three basic level concepts from which computer programming can be understood and discussed. The second section describes how embodied cognition can be used within a social semiotic framework, as a tool for analysing conceptual understanding from a cognitive linguistic standpoint; a theoretical position that is later expanded upon in the Methods section. The Results and Discussion section starts with an analysis of the data based on Grady’s Theory of Conceptual Metaphors described in the book *Philosophy in The Flesh* (1999), by George Lakoff and Marc Johnson, resulting in exposing nine different metaphors structured in accordance to bodily experiences. The findings are then discussed in relation to both the aims of the study, and the possible implications for computer science education.
Understanding Computer Programming

For students and teachers to be able to think about and understand computer science and computer programming, they need to develop an ability to talk and reason about aspects of programming, in a way that enables them to understand the difference between hardware and software, a variable and an attribute, or a function and a loop. Some of these differences can be found at a basic level, but some of them are much more difficult to identify. Of course, there are many different ways to understand computer science and computer programming but some are more common than others and serve as a foundation for a “common” understanding what computer programming is about. In this section I use examples found in literature (Stein 1999; Wing, 2006; Howland & Good, 2015; Jonsson et al., 2009; Reppening et al., 2011) to present three different ways of understanding computer programming at a basic level.

Almost all of the time, our brains are trying to organise and make sense of a vast amount of sensory information. Most of these activities occur unconsciously and together they shape the physical reality that we perceive (Lakoff & Johnson, 1999). It seems that the human brain has evolved to automatically place everything that we have around us into different mental categories, many of which are directly connected to sensory experiences and the way we use our bodies to interact with, and use them (Lakoff & Johnson, 1999). To use an example, I ask the reader to think of a chair. It has some of the same properties as a table – four equally long legs – but the big difference between a chair and a table lies in the way we interact with each respective object; we usually sit comfortably on our kitchen chairs, eating from a plate that is placed on the table. These basic differences in characteristics between a chair and a table are of significant importance in our understanding of the concepts “chair” and “table”, both in relation to ourselves and the rest of the world, and we can use these differences to organise simple concepts into different basic level categories (Lakoff & Johnson, 1999).

Using basic level categories, we can understand the difference between a cat and a dog or what separates a car from a motorcycle, and if it were not for our ability to blend, relate and understand categories in relation to others, our perception of reality would be very limited (Lakoff & Johnson, 1999). It is these connections between concepts that enable us to understand poetry, economics, arithmetic, gravity and global warming. The connections also allow us to talk about aspects of time, emotion and death (Lakoff & Johnson, 1999; Lakoff & Turner 1989).
The more one learns about a specific category, the more challenging it becomes to find simple and efficient ways to talk and think about concepts only using basic level categories. This is where our brains start to map previous experiences and understanding of one concept onto the properties of others. (Lakoff & Johnson, 1999). These cognitive mechanisms – conceptual metaphors – can take our understanding as giving life to non-existing concepts such as flying pigs or a bearded saint called Nicolaus. Hence, in this sense, it seems fair to suggest that these metaphors and categories play a large role in how we as human beings organise our surrounding in our attempts to understand the world (Lakoff & Johnson, 1980; 1999).

Stein (1999) refers to one “classic computational metaphor”, where computation is described as “a function from its inputs to its output … made up of a sequence of functional steps that produce – at its end – some result that is its goal” (Stein, 1999, p. 1). This is a widespread metaphor used in the field of computer science and is taken for granted by most computer scientists. What this computational metaphor expresses is that computation – almost exclusively – is a linear process owned by the programmer (Stein, 1999). However, the metaphor does not provide room for concepts also relevant to computer science, such as user interaction, or more complex concepts such as parallel processing. Another aspect not present is the use of modern developing tools or background services such as database communication. In other words, the computational metaphor – although still holding true for various aspects of computer programming – is not sufficient enough to cover many aspects of modern computer science. The metaphor in itself might not affect the skills of an experienced programmer, but if we adopt the premise that humans use metaphors as a way to organise knowledge, the computational metaphor has a big influence on computer education and the understanding of computer programming in general. The immediate result of this is that much-needed non-sequential computational thinking tends to be discouraged in schools, all in favour for a more linear way of thinking about computer programming (Stein, 1999). One example of this is found in Swedish schools where precise instructions and systematic instructions are central to programming (Skolverket, 2017).

A different way of understanding, talking and thinking about computer science and computation is to regard it as a way of thinking: “solving problems, designing systems and understanding human behaviour, by drawing on the concepts fundamental to computer science” (Wing, 2006, p. 33). According to Wing (2006), understanding computer science and computer programming is not understanding how computers ‘think’, work or what they ‘do’, it is rather an
understanding of how to “use our cleverness to tackle problems we would not dare take on before … limited only by our imaginations” (Wing, 2006, p. 35). This means, primarily, that computational thinking (CT) does not necessarily concern the understanding of every basic computational concepts, but rather the ability to communicate and conceptualise them effectively (Howland & Good, 2015). Although CT is now a widely accepted idea in K-12 programming education, it has received a colder reception within higher education, which is largely due to the complexity of the field computer science itself (Czerkawski & Lyman, 2015). For Czerkawski and Lyman (2015), computer science is more than just problem solving and programming – something that might be hard to grasp when relying exclusively on CT-education. To attain a deeper understanding of computer science and computer programming, Czerkawski and Lyman (2015) propagate the need for a vast repertoire of different didactic tools from many scientific disciplines.

Another aspect of computation, programming and/or computer science lies in the actual coding-process itself. The development of visually aided programming environments and open source libraries have made modern programming a far more social and material dependent activity. One reason for this is that the visual representations of the on-screen code evokes a spatial relationship between the programmer and the code, where hand gestures, facial expressions and even verbal communication become an integral part of the programming process. (Jonsson et al., 2009). This change actually reflects early programming psychology, which speculated that people’s ability to project themselves onto objects – mapping their bodily experiences onto the code – would help them overcome difficult programming problems (Reppening et al., 2014). Since graphical elements also can be used to represent abstract processes and statements, the visual approach to programming transforms computer programming and the conceptualization of code into a multimodal activity (Tversky et al., 2002), which in turn, induces a new set of conceptualise computer programming (Gallese & Lakoff, 2005).
Jonsson et al. (2009) also take on a performance perspective on programming when they assert “programming as a socially and physically situated activity, strongly shaped by the character of the available resources” (Jonsson et al., 2009, p. 118). By studying three different interaction settings from a performance perspective, the authors conclude that different settings – programming environments – call for different kinds of programming. There is evidently a connection between available and chosen media, and the way we use our bodies to interact with it. Albeit so, “further effort needs to be placed on understanding how new tools could be designed to support programmers of systems for extensive physical and bodily engagement” (Jonsson et al., 2009, p. 123).

Considering the basic level categories suggested above, there are no recipe-like or straightforward ways of understanding or formulating the essence of computer science and computer programming in simple terms. Instead, the common understanding of the field could, based on Lakoff and Johnson (1999), be organised into three basic level categories:

- a sequence of linear processes occurring within the computer itself (the computational metaphor)
- the way that the computer is used as a tool for problem solving (computational thinking)
- the meaning of the code, and the interaction with the software (the interaction with the code).

The above basic level categories do not differ from each other in the sense that they all concern the same type of activity. The difference lies rather in how the concepts “computer science” and “computer programming” relate to the scientist, programmer or in this case the students and the teachers. This exposes one of the challenges for communicating computer science in the classroom.

As stated above, categorising only on a basic level will highly limit our understanding of concepts. To further respond to the aims of the study, the next sections will describe a methodological framework for further identifying how teachers and students understand abstract aspects of computer science.
Methodological Framework – Using Social Semiotics for studying conceptual understanding in computer programming

Dating back to ancient Greece, doctors tried to understand human pathology by studying the manifestation of disease rather than its origin. The idea was that by observing the warning signs of the body, one would be able to trace – at least for that moment – unobservable disease (Danesi, 2007). The same idea – to understand the unobservable by exploring its manifestation – is central to social semiotics.

Semiotics – the study of the sign – has no real intrinsic theoretical assumptions or empirical methodologies (Chandler, 2007). The foundation of semiotics rests on the premise of identifying underlying, meaning making structures by using different theoretical frameworks as tools for analysing them. In this case, a sign could be a specific term in a large system of basic level concepts, but it could also concern artefacts such as graphical objects in a software, abstract programming semantics, classroom setup or even the facial expressions of the teacher. Using a semiotic framework as a tool for analysing the conceptual understanding in computer science means exploring the meaning associated with the signs used in computer science practices, in order to understand how conceptual understanding is grounded in everyday practice (Chandler, 2007).

Expanding into social semiotics, the study of social action, will include every type of sign there is. It is essentially a study of the verity in how signs are being used (Deely, 2006). From a social semiotic standpoint, communication is considered as being a social and multimodal act (Thylén et al., 2010; Lemke, 2010). It is concerned with the social meaning-making processes that take place in a present social environment. The social environment is to be considered multi-modal, meaning that it consists of much more than just verbal communication – although language plays a big role in how we evoke meaning into non-linguistic sings. The distinction between semiotics and social semiotics is important in this study, since it provides an opportunity to look beyond the sign itself, focusing on identifying the underlying processes of meaning making and understanding of a specific field.
Conceptual understanding manifested in language

Säljö (2014) terms the connection between communication and manifestation of a specific practice *mastering the semantics of a practice* – having the ability to talk about a practice, and also being able work within that particular practice. This is to say that if one is capable of building an aeroplane one will almost certainly be able to understand and communicate the fundamental physics of wing function. Likewise, if you can talk about how a computer works you will probably be able to use one. Lemke (1990) use the term *thematic-conceptual system* as “a … network of the relationships among the meanings of key terms in the language of a particular subject” (Lemke, 1980, p. 98). The thematic-conceptual system for, in this case computer programming, should not be understood as a pre-defined set of terms, valid in each community dealing with computer programming. The thematic-conceptual system is rather formed by the members of a specific practice in relation to their collective understanding of the subject (Lemke, 1980); the meaning of the code in relation to the dialogue around it.

For Säljö (2014) and Lemke (1990) a connection exists between our abilities to talk about something and to do something. In this way, language becomes our way of not only sharing our experiences with our peers, but it is also as a way to understand the world (Thylén et al., 2010). It follows, that language becomes our tool for putting ideas and thoughts out in the open; “a cultural construct that augments and enhances the natural potential of our bodies” (Thylén et al., 2010, p. 4). This makes it possible to explore conceptual understanding of computer programming as a relationship between the semantics of the code, and the way we talk about it while we are working with it (Visser, 2015).

Embodied Cognition as a tool for analysing conceptual understanding

In recent times there has been much work done on how humans are able to extract meaning from all that we perceive around us (Danesi, 2007). In doing so, cognitive linguists have started studying how we communicate and understand complex concepts based on how we use metaphors (Wilson, 2002). Consider the expression “I’m feeling blue today”. When read literally it would probably communicate the colour of your face or maybe the colour of your clothes, but when interpreted as a metaphor – linked to music – what it really does is project the emotional sadness of the blues onto your emotions. The associative structure offered by this metaphor is one way of understanding and conceptualising emotion in terms of music, many structures link meaning from one domain to another between signs (Danesi, 2007). By
identifying these structures, we are able to understand how we form meaning of the most abstract and complex sign systems.

In the seminal book, *Metaphors we live by* (1980), George Lakoff and Mark Johnson were amongst the first to study how we use our experience from one concept, in order to understand another. Their conclusion was that due to the way we use complex metaphorical language to communicate complex concepts, we also use the same metaphoric expressions to study the nature of the mind and how we evoke meaning into abstract concepts. Applying the works of Lakoff and Johnson to a social semiotic approach to explore conceptual understanding, transcends to exploring the use of metaphorical language within a specific social and scientific practice in order to understand the meaning of the thematic-conceptual system of the practice. By doing so, we are able to extract a qualitative description of how computer science and computer programming are understood as a “result of the way the brain and body are structured and the way they function in interpersonal relations and in the physical world” (Lakoff & Johnson, 1999, p. 37).

**Grounding of primary and conceptual metaphors in cognition**

Lakoff and Johnson (1980) introduce the term *conceptual metaphor*. Conceptual metaphors can be described as the “mappings across conceptual domains that structure our reasoning, our experience and our everyday language” (Lakoff & Johnson, 1999 p. 47). One example of this is the way we talk about quantities in ‘vertical’ terms, e.g. “interest rates are up” or, “raising the bar a bit higher”. According to Lakoff and Johnson (1999), the links between quantity and verticality is grounded in our everyday experiences, such as watching the level of water rise as we fill it with more water, while also realising that the water level will start to decrease at the same time we start pouring water from the glass. Sensory experiences like these give rise to the metaphor *MORE IS UP* (for reading purposes all examples of primary and conceptual metaphors will be notated with capital letters). Another example can be found in the way we understand time in terms of direction, e.g. “having tomorrow to look forward to” or, “leaving the past behind”. In this case, the conceptual bond between time, direction and movement is grounded in watching objects moving from one place to another at a specific speed during a specific amount of time (Lakoff & Johnson, 1999, Johnson, 2008). These sensorimotor experiences eventually form the associative structures that form the basis for our cognitive processes. This is evident in the plethora of metaphors that we use in language when uttering statements such as, “the temperature is rising”, “this is a mid-sized car”, “please lower your voice”, “you are
behind schedule”, and “if I could turn back time”. Without physically experiencing the correlation between “more” and “up” with our senses, we would be unable to understand the conceptual metaphors used in language, while the same reasoning applies for the connection between time and space.

When looking at other common expressions such as “I see what you mean”, it is apparent that we do not literally need to use our eyes in order to understand what someone is thinking. This rather shows that knowing is connected to the sensorimotor experience of seeing: SEEING IS KNOWING (Lakoff & Johnson, 1999). Such mapping has to do with the notion that looking at an object inside a box is the same as knowing that the object is inside the box. Since an object can also be perceived through our sense of touch, knowing can be understood as: TOUCHING IS KNOWING (Lakoff & Johnson, 1999, Lakoff & Turner, 1989). The same reasoning can be applied to the statement, “he is a warm person”. The metaphor reflects the connection we form between feeling safe and warm while being carried in a baby-blanket, hence the mapping between temperature and emotion: AFFECTION IS WARMTH (Lakoff & Johnson, 1999). However, and interestingly, it should be noted that this metaphor does not hold true in every part of the world - in areas where it is often too hot to carry babies in blankets, the connection between comfort and heat will not be made during early sensorimotor and bodily experiences.

Lakoff and Johnson (1999) argue that all conceptual metaphors manifested in language are based on bodily experiences and embodied in our brains. In other words, our metaphoric expressions are all embodied in both language and body. This means that our brain – analogous to the way we speak about one concept in terms of another – develops in relation to already embodied metaphorical concepts, leading to a reorganisation of the neurological pathways in our brains, thus creating new connections between pre-existing concepts and sensorimotor experiences and the mind (Lakoff & Johnson, 1999; Johnson, 2005; Gärtner, 2013). These processes lead to the formation of a more complex metaphorical network, expressed both in language and neural physiology. This theoretical underpinning forms the basis for what Lakoff and Johnson (1999) term embodied cognition.
From an embodied cognition perspective, “learning” is akin to embodying a new metaphor. Embodied cognition purports that our conceptual understanding is formed whilst interacting with the world, while at the same time altering it; we change reality, and at the same time reality changes us (Lakoff & Johnson, 1980). This implies that one learns from experiencing and interacting with reality, and the way in which one experiences the world is based on one’s own pre-existing conceptual understanding of it. Consequently, our conceptual understanding and cognition both exist in close relation with physical reality, as well with abstract constructions in our minds (Lakoff and Johnson, 1999; Gärtner, 2013). In this way, one’s already embodied metaphors become a reflection of one’s conceptual understanding. Hence, this opens a possibility for studying conceptual understanding – the embodied meaning – about a specific concept.

Since embodied metaphors are grounded in sensorimotor experiences, collecting and analysing data in relation to associated conceptual understanding requires observing and describing the basic sensorimotor experiences in which a conceptual metaphor is grounded. Such a methodological approach is adopted in the current study, and the following three theoretical tenets are stressed:

• Metaphors are by no means universal. Metaphors are grounded in our subjective sensorimotor experiences and judgements made in relation to ‘reality’ (Lakoff & Johnson, 1980, 1999). This means that our metaphors – our thoughts, reasoning and understanding of concepts – are neither subjective nor objective. Our understanding exists in relation to both reality and mind.

• Physical experiences are no more important than emotional, social or cultural experiences (Lakoff & Johnson, 1980). Even though we often talk about abstract concepts in terms of sensorimotor experiences, this is not to say that a feeling such as affection would not exist without experiencing warmth. On the other hand, the concept of affection would not be possible to understand without any embodied metaphors for affection (Lakoff & Johnsson, 1999).

• Everything we say is not grounded in metaphor, we often literally mean what we say (Lakoff & Johnson, 1999). For example, when we talk about “getting a good grip around the hammer” we literally mean what we are saying, but when we try to “get a grip on how a hammer can be used” we are linking gripping to understanding, as per the metaphor GRASPING IS UNDERSTANDING.
Johnson (2007) claims that “conceptual metaphor … is our primary (although not our only) means for abstract conceptualization and reasoning” (p. 179). Without conceptual metaphors, words would have no real meaning. Our understanding of concepts relies upon the use of metaphors. Without conceptual metaphors, our concepts would be nothing but semantic and grammatical structures, containing no meaning at all. According to Lakoff and Johnson (1999), our primary and conceptual metaphors are an integral part of our unconscious cognition, to the extent that no actual cognition can take place without the existence of metaphors. Taken together, this means that we need complex metaphors, not only to understand all the concepts surrounding us, but also to reason with and as a foundation for learning. Without our embodied metaphors, none of this would be possible.

The embodying of a metaphor in cognition

When Lakoff and Johnson (1980) first released Metaphors We Live By, they presented their thesis, more or less as a formal theory of linguistics. By the time they had released their second book on the topic – Philosophy in The Flesh – there had been a lot more research conducted in related areas, including neural modelling. In 1997, Srini Narayanan found that one would be able to model all neurological control for motion, based on the same type of basic neural structure (Lakoff & Johnson, 1999). Furthermore, Narayanan wanted to see if the model would be able to handle tasks other than movement alone. In his work, he found that his neural models could be applied in order to process more complex structures and abstract concepts such as international economics (Lakoff & Johnson, 1999). The results also show that there is a connection between the sensorimotor system in the brain and the understanding of complex metaphors. From a neural modelling perspective, Narayanan showed that a metaphor arises from the correlation between a sensorimotor operation (usually triggering movement or the detection thereof) and a subjective experience or judgement thereof. These types of operations trigger two separate “neurological” networks at the same time, which allows for the formation/embodying of novel metaphors.

We all form metaphors during everyday life – both in language and in the mind – each of them grounded in our subjective sensorimotor experiences and judgements of the world. The relatively simple metaphors consist of interconnected primary metaphors grounded in our everyday experiences. These metaphors are an integral part of how we understanding concepts and language (Lakoff & Johnsson, 1980, 1999). As soon as we are born, we involuntary start embodying primary metaphors just by observing the world around us (Lakoff & Johnson, 1999;
We feel the comforting warmth from our parents (AFFECTION IS WARMTH); we see the water level rising in a glass of water (MORE IS UP), and so forth. All of these sensorimotor experiences eventually culminate as constructed primary metaphors. Our attention to movement and spatial concepts are the most prominent aspects of cognition (Johnson, 2007). We unconsciously use spatial concepts to perceive, for example whether an object is located “near” or “far”, “inside” or “outside”. We also automatically use spatial concepts to properties such as entities moving “across” or “along” a specific path. We do so by activating two types of image schemas: the container schema and the source-path-goal schema (see figure 1) (Lakoff & Johnson, 1999). By placing concepts in a container schema, we are able to perceive what is inside or outside of a bounded region, enabling us to understand a butterfly being in the garden, or an algorithm working with a specific function of a software. The source-path-goal schema allows us to understand spatial relations and movement by attaining to a starting point, a path and an intended or unintended endpoint.

![Image](image.png)

**Figure 1**: An illustration of how a container schema and a source-path-goal schema relate to each other in a conceptual metaphor. By using the container schema as a vehicle of transportation along an imaginative path, we are able to understand the concept of movement.

While experiencing the world, we place primary and conceptual metaphors into mental image-schemas and frames (Lakoff & Johnson, 1999); some of which are embodied (grounded in sensorimotor experiences such as moving through physical space), and some of which are disembodied (grounded in everyday experiences such as being at the doctor’s office) (Lakoff & Johnson, 1999).
By placing our metaphors into an image-schemas and frames, we are able to construct meaning of what we perceive as our physical reality. By framing the metaphors, we also get a degree of foresight in understanding what will probably happen next in a sequence of particular events. Imagine for example, dining in a restaurant. When reading the menu, you might think, “the prices are higher than the last time you were there”. At this point, you are using a primary metaphor MORE IS UP placed in a source-path-goal schema. At the same time, your disembodied restaurant-frame tells you that the chef is to cook your ordered meal after the waiter has taken your order, not the other way around (Lakoff & Johnson, 1999). The restaurant frame is in turn an extension of a business-frame that tells you that you are about to receive a specific service in exchange for money. If we apply the same reasoning in thinking about solving a task related to computer programming, one might see something moving – a source-path-goal schema – on the screen, we can understand that the computer is processing some type of code. At the same time a factory-frame can tells us that transporting goods occurs from one place to another, we can use that to utter, “hold on a second, the computer is working”; a conceptual metaphor for data processing.

There is seldom just one specific experience serving as the basis for the more complex concepts. According to Grady’s theory of primary metaphors, our conceptual metaphors are formed by blending (integrating) different, simpler metaphors into a system of both new and already existing metaphors. In this system, a vast number of metaphors act together as a basis for reasoning and understanding particular concepts (Lakoff & Johnson, 1999). In his later work, Grady (2005) states that the concepts need to meet the following three criteria:

- One part of the conceptual metaphor must be sensory and one part must be non-sensory. The sensory part (sensory domain) will serve as the source for the metaphor and non-sensory part will serve as the target for the metaphor.
- Both parts of the conceptual metaphor must share the same schematic structure, meaning that for example both source- and target domain need to have the same spatial relationship (more-up) or actions (achieving a purpose-arriving at a destination).
- Both parts must covariate, meaning that an increase in at the source of the metaphor must result in an increase at the target.
These criteria will allow for conceptual blending to take place, where properties from the source domain are mapped on to the target domain, thus enabling us to understand one concept in terms of another.

**Analysing the embodiment of a conceptual metaphor in cognition**

The “grounding” of a complex metaphor does not occur in its whole, but rather in its individual component parts. This means that in order to understand the grounding of a complex metaphor, one needs to unravel its primary components, and subsequently metaphorically map its inherent parts back onto the metaphor; which in turn, serves as a tool for conceptualising the metaphor (Lakoff & Johnson, 1999).

Consider the metaphor **LIFE IS A JOURNEY**. Although many would label this a simple metaphor, it represents a multifaceted conceptual metaphor that maps the notion that we are supposed to achieve something in our lives, and that we during our lifetime are meant to work in different directions order to reach our goals. This statement contains two primary metaphors: **PURPOSES ARE DESTINATIONS** and **ACTIONS ARE MOTIONS**. These primary metaphors can be used to rewrite a metaphorical version of the statement as follows: You are meant to have a destination in life and you are meant to move in that direction. Continuing with the same logic – by combining the **LIFE IS A JOURNEY** metaphor with other complex metaphors – one can understand concepts such as feeling “lost in life” and “living in the fast lane”. In turn, the primary metaphors **PURPOSES ARE DESTINATIONS** and **ACTIONS ARE MOTIONS** are grounded in primary sensorimotor experiences that provide an understanding of the respective metaphor. So, in order to understand the conceptual metaphor **LIFE IS A JOURNEY** one needs to understand the sensory experiences in which both primary metaphors are grounded (Lakoff & Johnson, 1999). We often combine different complex metaphors to form ones that are even more complex. Consider someone saying, “It feels like I’ve reached a dead-end street!” What we know about a dead-end street (in itself a complex metaphor) effectively stops every journey imaginable, it lacks any direction, it stops any possibility of working towards a destination and the effort to reach the dead-end street has effectively gone wasted (Lakoff & Johnson, 1999).

It follows, that mapping our knowledge about the dead-end street onto the **LIFE IS A JOURNEY** metaphor provides us with an understanding about life, namely, that the metaphorical dead-end street hinders you from reaching your goals. It will not let you make progress in trying to reach your goal, it is to be understood as an obstacle in your life, and since it took a lot of effort to
reach the dead-end street, an equal amount of work has gone wasted. So, when someone is
talking about life in terms of a dead-end street, our knowledge about streets are effectively
being blended with the primary metaphors \textbf{PURPOSES ARE DESTINATIONS} and \textbf{ACTIONS ARE MOTIONS}, which in turn, have their own grounding primary experiences. By breaking down
contectual metaphors into primary metaphors, it becomes possible to chart the grounding and
mapping of a complex metaphor, thus making it possible to understand how we conceptualise
and understand abstract concepts such as computer science or computer programming. Specific
examples of conceptual metaphors related to computer programming will represent the
empirical findings of the study. The analysis adopts the above-described framework are
described next.

\textbf{A step-by-step description of the analysis}

Conducting this study is built upon the notion that cognition, reasoning and understanding
follow an inherent metaphorical logic (Lakoff & Johnson, 1980). Hence, it is possible to use
this logic to deconstruct and analyse any given conceptual metaphor in terms of its primary
parts. This is the foundation for analysing how the conceptual metaphors, expressed by the
respondents are formulated and embodied. The systematic method of analysis for reducing
complex metaphors into smaller primary metaphors is derived from Lakoff and Johnson (1999),
and comprises the following four overall steps:

1. Review excerpts of the interview transcripts from interviews (two teachers and three
   students working/studying at two mid-size upper secondary schools) with the aim to
   find passages that are rich in metaphorical language and where the student/teacher is
   interpreted by the researcher to utter some evidence of understanding of a particular
   concept.

2. Deconstructing the transcribed excerpts into their comprising primary metaphors,
   using examples of representative primary metaphors derived from Grady (1997).
   This will provide a basis for the metaphoric mapping and describe how each
   constituent primary metaphor are embodied. (Lakoff & Johnson, 1999).

3. Use the said primary metaphors to rephrase the initial statement into one or more
   metaphorical version of the statement, thus unveiling the mapping and logic of the
   initial statement (Lakoff & Johnson, 1999).

4. Map the metaphorical versions on to their sensorimotor experiences through
   application of Grady’s Theory of Primary Metaphor in order to reveal the
   sensorimotor experiences in which the whole statements are grounded.
Applying the above described method will present the embodiment of the uttered conceptual metaphors in a sequential manner. However, it is important to stress that this is only for notational reasons. From a neural perspective, conceptual metaphors arise from parallel neural connections and activity, grounded in its constituent parts (Lakoff & Johnson, 1999). This method does not provide for an explicit analysis of the neural processes of a conceptual metaphor, rather it serves as a basis for exploring the prerequisite thereof.

**Collecting information on how teachers and students understand aspects of computer programming**

The present study was carried out with three male students and two male teachers at two upper-secondary schools, located in different parts of Sweden. The respondents were selected based on a theoretical choice (Bryman, 2011) in order to uncover categories of conceptual metaphors, rather than testing hypotheses. Considering the complexity of computer programming and the study’s aim to explore a multitude of concepts, a choice was made to base the interview around fairly experienced students. Both teachers have a teachers-degree in programming and an educational background within computer science. The two teachers were contacted and asked to participate in the study. They were also asked to select suitable students for the project. All respondents have given their informed consent to participate in the study in accordance to Vetenskapsrådet’s (2017) *God forskningssed*. The information contained information regarding the aims and methods of the study. The respondents were also provided information on how and for how long the audio-recording would be stored.

The participants engaged in informal conversations with the researcher regarding different aspects of computer programming. The objective with this interactivist approach, where the interview is seen as a social activity, was to obtain a depth in the data than other types of interview techniques would allow (Ryen, 2004). The conversations were conducted in a largely unstructured manner by the researcher in such a way that the respondents decided which aspects to talk about without losing the focus on the general topic. This mutual participation in the interviewing process allow for gathering authentic insight into the experiences of the respondent in such a way that the data reaches a high degree of validity (Bryman, 2011). By not placing the study in an authentic educational environment there is a risk that information of social aspects in programming activities are lost. However, since this study aims to explore conceptual metaphors rather than specific practices, the setting are of less importance.
In preparation for the conversations, literature was reviewed in order to select relevant topics (e.g. software design, problem solving, computer code, the role of the programmer, etc.) for the study (Czerekawsky & Lyman, 2015; Lye & Koh, 2014; Wing, 2004; Stein, 1999). The conversations were audio-recorded (200 minutes of total recording time) using a portable device. Based on metaphorical content and topic, segments that involved metaphor usage and application of the conversations were transcribed and later translated into English. During the translation process, careful consideration was paid to preserving the maximum degree of initial metaphorical mappings conveyed in the respective Swedish excerpts.

The reason for focusing on the metaphorical content is the notion that our understanding of concepts is based on how small metaphorical units relate to form a larger whole, hence losing metaphorical content in the translation process would mean a decreased level of accuracy in the analysis of the data. As a consequence, only excerpts with a high degree of transferability between languages were included in this study.
Results and Discussion

Analysis of the data revealed the emergence of nine different conceptual metaphors related to aspects of computer programming and programming activities. Results of the study are framed by demonstrating the identification of each emergent metaphor. As described in the methodological framework, first, the utterances related to the datum in focus are presented. Second, primary metaphors (as notated in bold letters and numbered accordingly) are identified by using Grady’s theory of primary metaphor. Third, a metaphorical version of the statement is constructed. Fourth, the components of the metaphorical mapping are charted. Lastly, an analysis of the respective students’ and teachers’ use of the emergent metaphor in reasoning about computer programming are described.

The emergent metaphors are then presented in three main categories based on their respective sensorimotor domain. The metaphors are discussed in relation to sensorimotor experiences, image-schemas and frame, thus revealing a suite of primary metaphors that are engaged during computer programming activities.

Emerging Metaphors Regarding Computer Programming and Activities

Regarding a discussion around the interview question, ‘what does programming mean to you?’ student 1 begins to talk about the relationship between the programmer, the code and the client, as shown in the following excerpt:

“Student 1: Programming to me is being able to build¹ something that a user² can use to his or her advantage²⁴⁵… I would say that it’s like seeing³ how it is sitting on the other side and not understanding what is going on.”

The excerpt above holds five different primary metaphors, namely:

1. The programmer is building something (Organization Is Physical Structures)
2. The programmer has a relationship with the user (Relationships Are Enclosures)
3. The user has a purpose with placing the order (Purposes Are Desired Objects)
4. The programmer has to understand the purpose of the user (Knowing Is Seeing/Understanding Is Grasping)
5. The programmer has to work towards the purpose of the user (Purposes Are Destinations)
From this collection of primary metaphors, it is possible to restructure the primary metaphors into two metaphorical versions of the excerpt above. In the first version, programming is about taking different framework specific parts and organizing them into a meaningful whole. The other version states a cognitive aspect to programming, where the programmer has to be able to understand when the program has fulfilled its purpose of the client.

The two metaphorical versions can now be synthesised into the two following primary metaphors (presented in bold font), together with the respective image-schema (presented in italics). The corresponding subjective judgements, sensorimotor domains and primary experiences are charted in table 1 and 2.

i. **PROGRAMMING IS ORGANISING (PHYSICAL STRUCTURES)**

   *A person programming is an organiser*

   *The frame-specific program is destination*

   *Organizing is interacting with objects*

   *Table 1* shows the primary metaphors used to form the conceptual metaphor PROGRAMMING IS ORGANISING (PHYSICAL STUCTURE) and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstracting unifying</strong></td>
<td>Experience of Physical Objects</td>
<td>Interacting with complex objects and attending to their structure (correlation between observing part-whole structures and forming cognitive representations of logical representations)</td>
</tr>
<tr>
<td><strong>relationships</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Achieving a purpose</strong></td>
<td>Object manipulation</td>
<td>Grasping a desired object (correlation between satisfaction and holding a desired physical object)</td>
</tr>
</tbody>
</table>
ii. **PROGRAMMING IS SEEING**

*A person programming is a watcher*

*The user is the object for watching*

*The destination of the user is the destination for the programmer*

*Table 2* shows the primary metaphors used to form the conceptual metaphor **PROGRAMMING IS SEEING** and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Knowing</strong></td>
<td>Vision</td>
<td>Getting information through vision</td>
</tr>
<tr>
<td><strong>An interpersonal relationship</strong></td>
<td>Being in an enclosure</td>
<td>Living in the same enclosed physical space with the people you are most closely related to</td>
</tr>
<tr>
<td><strong>Achieving a purpose</strong></td>
<td>Reaching a destination</td>
<td>Reaching a destination throughout everyday life and thereby achieving purposes (e. g., if you want a drink you go to the water-cooler)</td>
</tr>
</tbody>
</table>

From the analysis above, Student 1 engages experiences from vision, movement in space and interaction with physical objects as the source domain for his conceptual metaphors (see table 1 and 2). In the first sentence, he is saying that programming is about *having the ability to interact and restructure different objects into part-whole structures and form logical representations of them until they reach a satisfactory correlation between them*. Using the logic of vision and movement, he is adding that programming is *observing when an object in the same enclosure is moving towards its designated place*. This means that – for him – programming is more than just designing a web site or creating an efficient way to communicate with a database, for example. He also includes the relationship between him – the programmer – and his future clients. Based on the above this means that computer science in practice is also a matter of empathy and social skill.

The two conceptual metaphors are framed in a way so that they correlate with each other; the better the structure, the quicker it will reach its designated place. This is one of Grady’s (2005) criteria for conceptual metaphors. This implies that both emergent metaphors serve as one complex metaphor for the programming process. Student 1 shows an example where his ability to write code – manipulating a structure, forming cognitive representations and logical
representations thereof – has led him towards understanding an underlying purpose for his programming; or vice versa. It is plausible that this type of complex understanding of programming can lead to an increased sense of meaning for the activity itself, and also an increased level of self-efficacy for the student.

**Emerging Metaphors Regarding Software Development**

Further on in the discussion, Student 1 converse about the programmer’s roll in software development, as depicted in the following excerpt:

“Student 1: To be able to develop (a software) you have to have an understanding\(^1\) for the knowledge of the client\(^2\), or how to explain it … so that you can meet with the client and deliver a product\(^3\) that he or she can understand\(^4\) and that is useable. Being able to place myself in someone else’s situation\(^5\). That is something that we have talked about a lot.”

The excerpt above holds five different primary metaphors:

1. The programmer needs an understanding for the knowledge of the client (*Understanding Is Grasping, Similarity Is Closeness*)
2. The programmer needs to meet the customer (*Change Is Motion, Intimacy Is closeness*)
3. The programmer has to deliver a product (*Change Is Motion, Purpose Is Desired Object*)
4. The customer needs to understand the product (*Understanding Is Grasping*)
5. The programmer needs to be able to understand the situation of someone else (*States Are Location, Understanding Is grasping*)

From this collection of primary metaphors, it is possible to restructure the primary metaphors into two metaphorical versions of the excerpts. In the first version the programmer – in order to understand the client – is a person who has to grasp the knowledge of the client. In the second version, the programmer is a person who has to deliver a product that matches the client’s ability to grasp the product.

The two metaphorical versions can now be synthesised into the following primary metaphor (presented in bold font), together with the respective image-schema (presented in italics). The corresponding subjective judgements, sensorimotor domains and primary experiences are charted in table 3.

---

\(^1\) Understanding
\(^2\) Knowledge of the client
\(^3\) Deliver a product
\(^4\) Understand
\(^5\) Place myself in someone else’s situation
i. **SOFTWARE DEVELOPMENT IS UNDERSTANDING (UNDERSTANDING IS GRASPING)**

*The programmer is grasping*

*The level of grasping of the customer is the object of the programmer*

*The satisfactory of the customer is the desired object for the programmer*

Table 3 shows the primary metaphors used to form the conceptual metaphor **SOFTWARE DEVELOPMENT IS UNDERSTANDING (UNDERSTANDING IS GRASPING)** and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comprehension</strong></td>
<td>Object manipulation</td>
<td>Getting information about an object by grasping and manipulating it</td>
</tr>
<tr>
<td><strong>Abstracting unifying relationships</strong></td>
<td>Experiences of physical objects</td>
<td>Interacting with complex objects and attending to their structure (correlation between observing part-whole structure and forming cognitive representations of logical relationships)</td>
</tr>
</tbody>
</table>

From the analysis above, Student 1 uses previous experiences of object manipulation as the source domain for his conceptual metaphors (see table 3). Using the logic of object manipulation, he is saying that software development is a process where *objects are restructured in such a way that it fits the hand of the client*. Following this conceptual metaphor, it becomes clear that the programmer’s ability to write code is, although necessary, secondary in software development.

This conceptual metaphor reveals a clear purpose for software development. The datum shows that he has made connections between the computer-frame and the business-frame, which exposes a complex conceptual understanding of computer programming.
Emerging Metaphors Regarding Algorithms

At one point in the interview with Teacher 1, a discussion on algorithms is brought to the fore, as communicated by the following excerpt:

“Teacher 1: Well, they are so simple … but that is maybe because I see them [the algorithms] as tools\(^1\). You must … well … let’s put it this way. This is a monkey wrench. This is a for-loop [a function that repeats an algorithm while a specific condition is true]. You have to kind of feel\(^2\) it and think about how to use it. It is not really a given thing how to use a monkey wrench either … But if you don’t know\(^3\) the tools then I don’t think that you will be able to solve the problems, because you have to connect\(^4\) the texts [in books or different assignments] to the available tools, and if you have understood\(^5\) the tools then maybe you can identify them in the text. “

The excerpt above holds five different primary metaphors:

1. An algorithm is a tool (Categories Are Containers)
2. The programmer has to have a feel for the tool (Purposes Are Desired Objects/Understanding Is Grasping)
3. The programmer has to understand what the tool can do (Change Is Motion/Actions Are Self-propelled Motions)
4. The programmer has to be able to make connection between text and tool (Purposes Are Destinations/Relationships Are Enclosures)
5. The programmer will be able to identify the algorithms in the text (Knowing Is Seeing).

From this collection of primary metaphors, it is possible to restructure the primary metaphors into one metaphorical version of the excerpts, where algorithms are confined sets of relationships between natural language and function. The programmer is gripping the relationships in order to understand them. In a second version, an algorithm is a tool - an object that is used to apply physical force.

The two metaphorical versions can now be synthesised into the two following primary metaphors (presented in bold font), together with the respective image-schema (presented in italics). The corresponding subjective judgements, sensorimotor domains and primary experiences are charted in table 4 and 5.
### Algorithm Is Grasping

*A person programming is the grasper*

*The container is the object*

*The container is the desired object*

*Table 4* shows the primary metaphors used to form the conceptual metaphor *Algorithm Is Grasping* and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehension</td>
<td>Object manipulation</td>
<td>Getting information about an object by grasping and manipulating it</td>
</tr>
<tr>
<td>Perception of kind</td>
<td>Space</td>
<td>Observing that things that go together tend to be in the same bounded region</td>
</tr>
<tr>
<td>Achieving a purpose</td>
<td>Object manipulation</td>
<td>Grasping a desired object</td>
</tr>
</tbody>
</table>

### Algorithm Is Physical Force

*A person programming is a controller*

*The algorithm is the object for control*

*The algorithm is moving an object into a specific direction*

*Table 5* shows the primary metaphors used to form the conceptual metaphor *Algorithm Is Physical Force* and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Being in control</td>
<td>Vertical orientation</td>
<td>Finding that it is easier to control another person or exert on an object from above where you have gravity working with you</td>
</tr>
<tr>
<td>Perception of kind</td>
<td>Space</td>
<td>Observing that things that go together tend to be in the same region</td>
</tr>
<tr>
<td>Achieving results</td>
<td>Exertion of force</td>
<td>Achieving results by exerting force on physical objects to move or change them</td>
</tr>
</tbody>
</table>
From the analysis above, Teacher 1 uses experiences of touch as the source domain for his conceptual metaphor. Using the logic of touch, he describes algorithms as something tactile that a programmer is supposed to hold and compare to natural language. When using the tool-analogy, Teacher 1 is using movement as the source domain for his conceptual metaphor. The algorithm then becomes an item that is used to apply physical force onto an object in order to set the object into motion.

Attention to movement is one of the most fundamental dimensions to language and cognition. As humans, “we are born moving. It is originally through movement that we come to inhabit a world that makes sense to us” (Johnson, 2007, p. 20). In order to understand movement, we must go beyond the structures of movement by attending to the quality of motion (Johnson, 2007). Johnson (2007) states three necessary criteria for fully understanding movement: tension (the amount of force needed to perform an action), linearity (the path and trajectory that the object will follow during motion) and amplitude (the manner in which we perform an action). Furthermore, another necessary criterion for understanding motion is that movement has to take place in a physical space; the context of motion. Applying Johnson’s criterion on the conceptual metaphors above shows that, in order to fully understand algorithms, one must get a “feel for” both the quality of the loop (the efficiency, how it functions, where it will move objects, etc.) and the context in which it is operating (the relationship between natural language and the semantics of the code). It follows, that this understanding is seen as providing one with the power that you need to control a computer. Programming in the excerpt above is expressed as using the power of the code (language) to take control of every process going on within the computer itself. To do so, the programmer has to understand the connections between natural language and the semantics of the code.
Emerging Metaphors Regarding Loops

A way to solve complex programming problems is through the use of loops (small sections of code that are iterated a number of times). In the following transcript extract, Student 2 provides an explanation of what a loop is and what it might do:

“Student 2: An iteration is really a loop, and then it is that you work¹ through a code many times to get a lot of data of the same or different kind². But then we are going to work with objects where we get data from other parts of the code instead of running it line by line³ so to say. Instead of running it line by line we are sort of picking apples⁴ from a tree.”

The excerpt above holds four different primary metaphors:

1. An iteration is working through the code with the purpose to obtain values (Purposes Are Desired Objects/Purposes Are Destinations)
2. There are different kinds of values (Categories Are Containers)
3. The contradictory approach to using loops is running a linear process (Linear Scales Are Paths/Change Is Motion/Actions Are Self-propelled Motion)
4. Using loops is like picking apples from a tree (Understanding Is Grasping/States Are Location/Purposes Are Destinations/Organization Is Physical Structure)

From this collection of primary metaphors, it is possible to restructure the primary metaphors into two metaphorical versions of the excerpt. In the first version, the loop is used interchangeably with iteration. A metaphorical version of this can be stated as the loop being a self-propelled linear motion a number of times to move a desired object of some category to a desired destination. The second version adds a purpose to the loop. A metaphorical version of this can be stated as the loop having the purpose of collecting desired objects (structures) found along a linear path and moving the structures into another structure.

The two metaphorical versions can now be synthesised into the two following primary metaphors (presented in bold font), together with the respective image-schema (presented in italics). The corresponding subjective judgements, sensorimotor domains and primary experiences are charted in table 6 and 7.
i. **LOOP IS PURPOSE**

* A loop is a cause
* A loop is moving categories from one place to another

*Table 6* shows the primary metaphors used to form the conceptual metaphor **LOOP IS PURPOSE** and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Achieving results</strong></td>
<td>Exertion of force</td>
<td>Achieving results by exerting forces on physical objects to move and change them</td>
</tr>
<tr>
<td><strong>Achieving a purpose</strong></td>
<td>Reaching destination</td>
<td>Reaching destinations throughout everyday life and thereby achieving purposes</td>
</tr>
<tr>
<td><strong>Perception of kind</strong></td>
<td>Space</td>
<td>Observing that things that go together tend to be in the same bounded region</td>
</tr>
</tbody>
</table>

ii. **LOOP IS PHYSICAL STRUCTURE**

* A loop is grasping desired objects
* A loop moves objects from one state to another
* A loop changes the physical structure of the code

*Table 7* shows the primary metaphors used to form the conceptual metaphor **LOOP IS PHYSICAL STRUCTURE** and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comprehension</strong></td>
<td>Object manipulation</td>
<td>Getting information about an object by grasping and manipulating it</td>
</tr>
<tr>
<td><strong>Perception of kind</strong></td>
<td>Space</td>
<td>Observing that things that go together tend to be in the same region</td>
</tr>
<tr>
<td><strong>Abstracting unifying relationships</strong></td>
<td>Experiences of physical objects</td>
<td>Interacting with complex objects and attending to their structure (correlation between observing part-whole structure and forming cognitive representations of logical relationships)</td>
</tr>
</tbody>
</table>
From the analysis above, student 2 uses experiences of physical force and movement in space as the *source domain* for his conceptual metaphors (see table 6 and 7). Using the logic of motion, he is saying that a *loop pushes something from one place to another inside the structure of the program*. Inside the loop – the loop now being the object that is to be moved – there exists some kind of information (data) that will somehow move along the code (program). When breaking down Student 2’s statement it is clear that the inherent logic of this network of primary metaphor is erroneous. He frames the loop-concept as being both a verb and a noun at the same time.

Revisiting Lakoff and Johnson (1999), one can see why the conceptual metaphor fails in this instance - the loop is framed in both a container and a source-path-goal schema, at the same time. From an embodied cognition perspective, this is impossible, since an object cannot be inside a container while traveling along itself. Albeit so, when comparing student 2’s metaphorical use with Stein’s (1999) metaphor for computation, there are some interesting resemblances. The metaphor for computation states that, “Computation is a function from its inputs to its output” (Stein, 1999, p. 1); here, computation is something that is being pushed through the program. “It is made up of a sequence of functional steps that produce—at its end—some result that is its goal”. In this regard, computation now becomes a set of functions (objects) that will eventually reach some destination. Student 2’s reasoning indicates the inadequacy of Stein’s (1999) metaphor for computation. It holds for programs that run in a linear fashion, but it omits parallel processing. Also, it is only viable if one frames the function concept in a source-path-goal schema.

Another aspect in student 2’s reasoning is his lack of contextual content. Neither the programmer nor the code is really connected to the loop concept or the use of loops. Observations as such are common in computer science education, where programming concepts are rarely discussed in an authentic manner (Weintrop & Wilensky, 2014).
Emerging Metaphors Regarding Different Types of Students

As part of an interview with Teacher 2, he provides utterances related to how different students address different types of problems, as shown in the following excerpt:

“Teacher 2: There are those who look at the tools\(^1\) (software development tools) and say that it is much cooler to build\(^1\) your own tool… and then they start building from scratch\(^2\). … They want to create a better tool, and if they continue within the world of programming\(^3\) they usually end up as engine\(^4\) developers.”

The excerpt above holds four different primary metaphors:

1. A tool is something you either a pre-packed function or something you build on your own (Purposes Are Desired Objects/Organizing Is Physical Structure/Knowing Is Seeing)
2. Building something starts at zero (More Is Up/Linear Scales Are Paths)
3. For these kind of students, continuing within computer science means moving towards a specific interest in computer science (Time Is Motion/States Are Locations/Change Is Motion/Relationships Are Enclosures)
4. Computation is in its basic form a self-propelled motion (Actions Are Self-Propelled Motions)

From this collection of primary metaphors, it is possible to restructure the primary metaphors into two metaphorical versions of the excerpts. The first version states that programming is choosing between building their own structures or using prefabricated ones. The second version states that tool building is about understanding connections between categories in relation to a specific problem.

The two metaphorical versions can now be synthesised into the two following primary metaphors (presented in bold font), together with the respective image-schema (presented in italics). The corresponding subjective judgements, sensorimotor domains and primary experiences are charted in table 8 and 9.
i. **PROGRAMMING IS DESIRED OBJECTS**

A person programming is using or building tools (organizing structures)
The tool is the object
The container is the desired object

*Table 8* shows the primary metaphors used to form the conceptual metaphor **PROGRAMMING IS DESIRED OBJECTS** and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstracting unifying relationships</strong></td>
<td>Experience physical objects</td>
<td>Interacting with complex objects and attending to their structure (correlation between observing part-whole structure and forming cognitive representations of logical relationships)</td>
</tr>
<tr>
<td><strong>Achieving purpose</strong></td>
<td>Object manipulation</td>
<td>Grasping a desired object (Correlation between satisfaction and holding a desired physical object)</td>
</tr>
<tr>
<td><strong>Perception of kind</strong></td>
<td>Space</td>
<td>Observing that things that go together tend to be in the same bounded region (correlation between common location and common properties, functions or origins)</td>
</tr>
</tbody>
</table>
ii. **PROGRAMMING IS SEEING**

*Different tools are being used to solve a problem*
*The tools all have their own inherent qualities that are useful for the programmer*
*A person programming is an observer of actions in relation of the tools that are being used*

*Table 9 shows the primary metaphors used to form the conceptual metaphor PROGRAMMING IS SEEING and how they are grounded in sensorimotor experiences (Lakoff & Johnson, 1999).*

<table>
<thead>
<tr>
<th>Subjective judgement</th>
<th>Sensorimotor domain</th>
<th>Primary Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
<td>Moving your body thorough space</td>
<td>The common action of moving yourself through space, especially in the early years of your life</td>
</tr>
<tr>
<td><strong>A subjective state</strong></td>
<td>Being in a bounded region of Space</td>
<td>Experience a certain state as correlated with a certain location</td>
</tr>
<tr>
<td><strong>Knowledge</strong></td>
<td>Vision</td>
<td>Getting information through vision</td>
</tr>
</tbody>
</table>

Based on the analysis above, it is evident that Teacher 2 uses experiences of movement and object manipulation as the source domain for his conceptual metaphors. Using the logic of object manipulation and movement, he is saying that students tend to either *interact with objects and attend to their structures or find satisfactory correlations between objects of different kinds*. Using the logic of movement, he is saying that in order to build better tools (structures) for programming, a programmer has *to move different structures through space and see how a specific state (in this case function) correlates with a location*.

In this excerpt, teacher 2 is mainly talking about the type of students who are able to make a quick judgement whether to use pre-packed tools to solve a problem, or to organise their own objects in order to reach their purpose. The emerged conceptual metaphors imply that teacher 2 finds that the abilities required for this choice rest upon being able to understand code as a metaphor for a tool that is moving and/or restructuring different objects into the right shape and the right state/position, using visual concepts to decide whether the program code is efficient or not.
In practice, Teacher 2 is saying that, in order to either choose or build a tool (code), the programmer needs to be able to find unifying factors of specific objects (algorithms, functions, etc.) and desired objects (specific problems that are to be solved). To do so, the programmer requires the ability to identify similarities between objects and structure them in a part-whole hierarchy. The programmer is then required to control the tools in such a way that all available objects are connected so that they can move in the same direction. This conceptual metaphor is a far more complex way of describing and understanding programming, than Stein’s computational metaphor (1999), but it would be one way of expressing the concept of computational thinking. In one sentence, Teacher 2 is able to describe his own version of Wing’s “our cleverness to tackle problems we would not dare take on before” (Wing, 2006, p. 33).

Johnson et al. (2009) states that different settings call for different kinds of programming. This implies that, depending on the choice of tool or problem to be solved, there would be a need for a different media and/or classroom configuration. One example of a situation like this is exposed when teacher 2 says, “There are those who look at the tools (software development tools) and say that it is much cooler to build your own tool”. By allowing the student decide on which, or how different tools are to be used for problem solving, teacher 2 is giving them a possibility to embody their knowledge in different ways.

In a more general discussion about classroom settings and teaching methods, teacher 2 talks about the use of drawings, group discussions, and one-to-one dialogue as didactic tools in the classroom. In this way, he offers a restructuring of the social practice setting in the classroom. This indicates that there are pragmatic connections expected between the metaphoric language of Teacher 2 and the actual learning environment of the classroom.
Reflecting on the emergent programming metaphors as a collective

The analysis has revealed various common metaphors across the data corpus. However, when taking the grounding and framing of the metaphors into consideration, a whole new understanding of the teacher’s and student’s understanding of computer programming appears. In this section I will structure the different primary metaphors and the corresponding image-schemas and frames in relation to their respective sensory domains. The primary metaphors are presented in bold letters with their respective image-schema in italic. This will be used to discuss how the different primary metaphors are used by the teachers and the students, thus unveiling their conceptual understanding of computer programming.

**PROGRAMMING IS SEEING**

1) *A person programming is a watcher*
   *The user is the object for watching*
   *The destination of the user is the destination for the programmer*
   *The enclosure between the users is in the destination.*

2) *Different tools are being used to solve a problem*
   *The tools all have their own inherent qualities that are useful for the programmer*
   *A person programming is an observer of actions in relation of the tools that are being used*

Both Student 1 (image-schema 1) and Teacher 2 (image-schema 1) use metaphors grounded in visual experiences, but there are a few clear differences in how they frame their respective conceptual metaphors. Both Teacher 2 and Student 1 place programming in a source-path-goal schema, meaning that programming is understood as a process. Student 1 is framing the programming process in a “business-frame”, where the programmer is a seller and the user is a buyer. In order to attain a reasonably good business relationship, the programmer has to design a product (code) with an ambition to solve a problem for the customer. In this sense, the motivation for learning programming can be found in the ambition to satisfy customers’ expectations, emphasising a need for abilities such as communication and empathy in computer science education.

As for Teacher 2, seeing is framed in a specific situation where a programmer has to decide how to solve a specific problem. With the vast variety of visual development tools available today, a programmer is given possibilities to use a larger array of sensorimotor experiences to embody code. One example of this is SCRATCH, where movement is central to the user. Another example is how Teacher 2 uses group discussions in his teaching, instead of focusing
on typing code. As Howland and Good (2015) point out, there is nothing saying that an “unskilled” programmer should not be able to see the solution to a programming problem, but without the ability to evaluate which solution to use, it becomes apparent why he or she would have a hard time finding the best one. This goes hand in hand with the conceptual metaphor of Teacher 2, where the relation between tool, problem and quality are all of importance.

**PROGRAMMING IS GRASPING**

3) *The programmer is grasping*
   *The level of grasping of the customer is the object for the grasping of the programmer*
   *The satisfactory of the customer is the desired object for the programmer*

4) *A person programming is the grasper*
   *The tool is the container*
   *The container is the desired object*

The sensory domain for grasping is object manipulation. It is grounded in the notion of obtaining information from an object itself and serves as a primary metaphor for knowing (Lakoff & Johnson, 1999). In other words, these two conceptual metaphors are related to knowledge: knowledge about the customer and knowledge about programming.

While still framing his conceptual metaphor in a “business-frame”, the main interest for Student 1(image-schema 3) is the customer. The conceptual metaphor also reveals the one specific aspect of interest - the knowledge of the customer. Student 1 understands that in order to succeed as a software developer, one has to develop the ability to adapt to the customer’s level of understanding. Programming becomes a way to help other people in their work or everyday life, and as a programmer, Student 1 sees himself as a part of that process. Another important aspect of this metaphor is the fact that it is framed outside a formal educational environment.

As for Teacher 1 (image-schema 4), the object to grasp is the available tools in the programming process, where different tools have different properties. This metaphor might lead to a quantitative understanding of computer science, where tool after tool are being taught, yet not necessarily being put into context. On the other hand, when combined with the previous metaphor, the knowledge about the tools becomes an important part of the problem-solving process in programming.
PROGRAMMING IS PHYSICAL STRUCTURE

5. A person programming is an organiser
   The frame-specific program is destination
   Organizing is interacting with complex objects

6. A loop is grasping desired objects
   A loop moves objects from one state to another
   A loop changes the physical structure of the code

Student 1 (image-schema 5) uses organizing as a metaphor for a computer program. Programming to him is much like moving objects. As a programmer, he is expected to arrange these objects in a desired pattern in order to obtain a desired function. His previous metaphors have been used in a business-frame, emphasizing the relationship between the programmer and the customer. In using this metaphor, he is focusing on the relationship between the programmer and the computer, hence defining his role as an engineer. Programming is now a relationship between him and the code.

As for Student 2 (image-schema 6), the metaphor lacks a connection between the programmer and the code. This conceptual metaphor, where code is something that exists on its own, without any purpose makes it impossible for Student 2 to develop a more complex understanding of programming. However, from the analysis, it is possible to understand what aspects of computer science he needs to develop further.

PROGRAMMING IS MOVEMENT

7. A person programming is a controller
   The algorithm is the object for control
   The algorithm is to move an object into a specific direction

As shown in the extract above, while talking about algorithms, Teacher 1 (image-schema 7) uses metaphors related to movement. For him an algorithm is an object that can move another object – via a source-path-goal schema – in a specific direction. Using algorithms, the programmer is able to control the program and all of its functions in the right direction. This metaphor states that learning to program means developing the ability to foresee the way in which algorithms are going to control the data processing and how the program are supposed to handle interactions with the computer.
Conclusions and Implications

The aim of this study has been to: firstly, explore how conceptual metaphors are used by students and teachers engaging in conversations about computer programming; secondly, use the emerged conceptual metaphors to identify how teachers and students understand abstract aspects of computer programming and thirdly, identify the sensorimotor experiences shaping the conceptual metaphors used by the students and the teachers. This has been performed by analysing transcripts of five different conversations on the topic. Grady’s Theory of Primary Metaphor has been used as the primary analytical tool for the analysis. As a result, five statements have been deconstructed and grouped according to their respective sensory domain.

This study suggests that there is more to understanding computer science than just understanding the typing of code and the function of the computer. Understanding computer science concerns human interaction and relations as part of a complex and multifaceted meaning-making process. There are also important aspects of identifying visual similarities and common features of different kinds of objects. Furthermore, computer science concerns questions about societal understanding, customer service, and empathy. Finally, this study indicates that a large component of understanding computer science has to do with the interpersonal relationship between the programmer and the computer, which includes an understanding of semantics of the code. Another interesting dimension is that the respondents did talk about the importance of computer science not only for themselves, but also in terms of how their knowledge will come to affect a specific social practice and society in general.

The majority of metaphors identified in this study are mapped to different sensorimotor experiences, making it apparent that computer science and computer programming should be considered, and taught as a multimodal process. The coding process in itself is only a small part of acquiring a basic understanding of computer science. Applying the results of this study in practice would mean designing a learning environment where all of these sensorimotor experiences are at least taken into consideration. One example of this might be finding ways to help students map the sensorimotor experiences of organizing objects on to the more abstract sorting algorithms found in computer science. Another example would be to create a learning environment that encourages dialogue – both peer-to-peer and student-to-teacher.
Since conceptual metaphors often correspond to more than one sensorimotor experience, they are also grounded in more than one of our senses. This is what Lakoff and Johnsson (1999) terms the aesthetics of the metaphor. This might be a basis for understanding the multimodal perspective of computer science. In this study, it is obvious that the conceptual understanding shown by the respondents is far more complex than being grounded in mere a single sensorimotor experience of interacting with a computer or reading code in a book. Rather, it is a blend of tactile, visual, spatial experiences that leads to a better understanding of computer science. This means that by increasing the use of multimodal didactic tools such as different types of visual developing software could become an important resource for enhanced learning in the classroom.

Based on the sensory domain of the respective conceptual metaphor, the results of this study suggest that by promoting the following basic sensorimotor experiences would have a positive effect on how learners are able to conceptualise different aspects of computer programming:

- Interacting with complex objects and attending to their structure (correlation between observing part-whole structures and forming cognitive representations of logical representations)
- Grasping a desired object (correlation between satisfaction and holding a desired physical object)
- Reaching a destination throughout everyday life and thereby achieving purposes (correlation between engaging in an activity and finding it satisfactory)
- Achieving results by exerting force on physical objects on order to move, rearrange, change them
- Obtaining information about an object by grasping and manipulating it and observing that things that go together tend to be in the same bounded region (correlation between common location and common properties, functions or origins)
- The common action of moving yourself through space, especially in the early years of your life is vital for embodying the majority of primary metaphors.
By designing learning situations based on the above sensorimotor experiences, teachers can combine different types of learning activities to promote a more diverse understanding of computer programming. Furthermore, this way of developing conceptual understanding could increase the amount of available conceptual metaphors, which would expand the possibilities for more faceted learning experiences in computer programming.

Acknowledgements

I would like to acknowledge the teachers and students that participated in the study. I would also like to acknowledge Maria Sparf for valuable comments about the text and Anders Jidesjö for introducing and guiding me through the first stages of this project. Lastly, I would like to thank my supervisor Konrad Schönborn for endless support, inputs and comments during the entire project. I could not have done it without you.
References


