Nelson Mandela Metropolitan University

for tomorrow

Department of Computing Sciences

A Natural User Interface architecture using gestures to facilitate the detection of Fundamental Movement Skills

Richard Amanzi

Supervisor: Prof. J. Greyling
Co-Supervisor: Prof. C. Cilliers
Department of Computing Sciences

December 2014

Submitted in fulfilment of the requirements for the degree of
Magister Scientiae in the Faculty of Science at the Nelson Mandela Metropolitan University

Declaration

I, Richard Amanzi, hereby declare that the dissertation for the degree Magister Scientiae is my own work and that it has not previously been submitted for assessment or completion of any postgraduate qualification to another University or for another qualification.

Richard Amanzi

Acknowledgements

I would like to thank my supervisors, Prof. Jean Greyling, and Prof. Charmain Cilliers, for their support and continual guidance for the duration of this research. I wish to express my gratitude for the hours spent reading through this document and the suggestions that were made to improve the content and structure.

I would also like to thank NMMU for the financial assistance provided during the course of this research study. Sincere appreciation to the Department of Computing Sciences for the resources they provided. Special thanks to my family and friends whose continued love and support helped me complete this work.

Table of Contents

Summary

Fundamental movement skills (FMSs) are considered to be one of the essential phases of motor skill development. The proper development of FMSs allows children to participate in more advanced forms of movements and sports. To be able to perform an FMS correctly, children need to learn the right way of performing it. By making use of technology, a system can be developed that can help facilitate the learning of FMSs.

The objective of the research was to propose an effective natural user interface (NUI) architecture for detecting FMSs using the Kinect. In order to achieve the stated objective, an investigation into FMSs and the challenges faced when teaching them was presented. An investigation into NUIs was also presented including the merits of the Kinect as the most appropriate device to be used to facilitate the detection of an FMS. An NUI architecture was proposed that uses the Kinect to facilitate the detection of an FMS. A framework was implemented from the design of the architecture. The successful implementation of the framework provides evidence that the design of the proposed architecture is feasible. An instance of the framework incorporating the jump FMS was used as a case study in the development of a prototype that detects the correct and incorrect performance of a jump. The evaluation of the prototype proved the following:

- The developed prototype was effective in detecting the correct and incorrect performance of the jump FMS; and
- The implemented framework was robust for the incorporation of an FMS.

The successful implementation of the prototype shows that an effective NUI architecture using the Kinect can be used to facilitate the detection of FMSs. The proposed architecture provides a structured way of developing a system using the Kinect to facilitate the detection of FMSs. This allows developers to add future FMSs to the system. This dissertation therefore makes the following contributions:

- An experimental design to evaluate the effectiveness of a prototype that detects FMSs
- A robust framework that incorporates FMSs; and
- An effective NUI architecture to facilitate the detection of fundamental movement skills using the Kinect.

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

Fundamental Movement Skills (FMSs) are movement skills that are observed in children between the ages of two and seven years as they gain control over their gross motor skills. These skills are essential for children to successfully participate in a range of sports and allow them continued involvement during childhood and adolescence (Lubans, Morgan, Cliff, Barnett, & Okely, 2010). Movement skills taught in early childhood, such as running, jumping, leaping, and throwing, form the building blocks for more advanced movements used in sports such as basketball, cricket, javelin and netball (Okely, Booth, & Patterson, 2001). Children taught these skills early in their childhood are more likely to live a physically active lifestyle and therefore reduce their chances of being overweight and obese (Williams, Carter, Kibbe, & Dennison, 2009).

Children do not acquire FMSs during maturation as is commonly believed, but through instruction and practice (Haywood & Getchell, 2009). The development of movement skills in children varies with each child developing at his/her own rate. The children who are provided with structured opportunities by teachers to practise and learn FMSs develop confidence and proficiency in FMSs. A variety of playground equipment such as foam balls, jump ropes and balance beams helps to promote the development of FMSs when they are made available for children. The desire to move shown by children compels them to take part in developmental games that can be set up by teachers using playground equipment. Thus tools play an important support role when children learn FMSs.

With the introduction of the Nintendo Wii Remote (Nintendo Entertainment system, 2013), Sony's Move controller (Sony Computer Entertainment, 2013) and Kinect (Microsoft Corporation, 2013a), users are able to directly control game play with their body or parts thereof. In the case of the Wii, new motion capabilities allow users to use realistic physical actions to swing virtual golf clubs, roll virtual bowling balls, punch out virtual enemies, and point directly to menu items on the screen by moving the hand-held controllers. The Sony Move controller is similar to the Nintendo Wii but allows for more precise forms of 3D interaction by determining the absolute 3D position and orientation of the Move controller (Bowman, McMahan, & Ragan, 2012). The Kinect allows users to control movement by using their entire bodies.

This recent evolution of video game controllers allows players to interact with games called exergames or active games. These exergames can help children engage with games physically and socially. The exergames can be used to simulate playground equipment by allowing children to perform movements they would normally perform using equipment like foam balls and jump ropes. Using these technologies, it is possible to track movement of users and encourage the mimicking of FMSs.

A technological solution could therefore, be considered as an option to assist in the instruction and assessment of the necessary movement skills in children as an aid for teachers. A device like the Wii, Move or Kinect could be used as a tool to assist in the detection of FMSs. Children could attempt to perform an FMS and the device could detect the accuracy of the movements made. This would provide an opportunity for children to learn FMSs by giving them feedback on their performance.

1.2 Relevance of Research

Barnett, Morgan, Van Beurden, and Beard (2008) express concern that the current level of FMSs evident in younger children and many adolescents would be insufficient to allow engagement in structured team or individual sports as they have not had sufficient opportunities to practise and develop them. Lack of development of these skills early in childhood leads to difficulties in mastering more advanced movement skills, which may cause frustration for children and result in them leading a less physically active lifestyle and consequently increasing the level of fat in their body resulting in subsequent obesity (Williams et al., 2009).

Children require explicit teaching in order to develop proficiency or mastery of FMSs (Gallahue & Ozmun, 2002). The immediate surrounding areas and space that can be occupied by children, allow them to attempt to learn movement skills. These surroundings should include opportunities for practice, encouragement, and instruction because they are vital for the proper development of FMSs (Payne & Isaacs, 2002). The teachers assigned to teach FMSs give encouragement and instruction and provide the children with opportunities for practice. The tools they use to teach children include playground equipment and these are allocated according to the number of children who can be instructed by the teacher (Gallahue & Donnelly, 2003). For children to achieve proficiency and mastery of FMSs, they must be given instructions of how to perform FMSs and given opportunities to practise. Booth et al. (1999) suggest that sufficient curriculum time and suitably qualified and resourced staff need to be made available to support FMSs development among primary school children.

The learning of these skills is dependent on the expertise of the teacher, available equipment and the complexity of the skill to be learned (Booth et al., 1999; Department of Education, 2009). One of the challenges is that many teachers are not sufficiently skilled in the instruction and assessment of FMSs (Lubans et al., 2010; Matin & Hands, 2003). Furthermore, there is not enough time in the school curriculum for the development of FMSs. It takes a significant amount of time to teach children to perform FMSs correctly. Children are required to have 60 minutes of dedicated physical activity per day and most of this time is occupied by free play (Gagen & Getchell, 2006). Current intervention programs show improvements in FMSs in children when teachers are provided with the tools such as playground equipment for improving these skills. It is clear that when teachers are supported with tools to teach FMSs, there is a significant improvement of these skills in children (Mitchell et al., 2011).

1.3 Aim of Research

The aim of this research is:

To propose an effective natural user interface architecture, using the Kinect to facilitate the detection of fundamental movement skills.

1.4 Research Objectives

In order to achieve the aim of the research, this research study will seek to attain the following objectives:

1. To investigate and describe the types of fundamental movement skills and the challenges in teaching them;

2. To propose an NUI architecture to be used in the detection of fundamental movement skills using the features of the Kinect; and
3. To apply and evaluate the proposed architecture.

1.5 Research Questions

The main research question to be addressed by this research is:

How can a natural user interface architecture for Kinect be designed that can effectively facilitate the detection of fundamental movement skills?

The following sub-questions will be used to answer the main research question given above:

1. What are fundamental movement skills?
2. What are the challenges of teaching and learning fundamental movement skills?
3. How can Kinect support a natural user interface to detect fundamental movement skills?
4. How is the Kinect used?
5. How can a natural user interface architecture be designed?
6. How can a prototype of a Kinect be implemented by applying the architecture?
7. How effective is the prototype in the detection of a fundamental movement skill?

1.6 Scope and Constraints

The scope of this research will be limited to the design of an NUI architecture that can be used for detecting FMSs. The designed architecture should facilitate the adding of FMSs supporting plug and play. To evaluate the designed architecture, a framework will be implemented, by applying the architecture. The framework will incorporate an instance of a single FMS as a case study. The developed prototype will be a natural user interface (NUI) application that should be able to track the user and detect whether an FMS is performed correctly or not. It should give feedback to users by displaying the recorded output of the performed skill. The NUI will be developed to track the movements of only one user when detecting FMSs.

FMSs will be limited to gross motor skills and the development of these skills in early childhood. The movement to be implemented will involve a single selected FMS.

1.7 Research Methodology

This research aims to produce an effective solution that can be used to facilitate the detection of FMSs. It will follow a goal-directed approach through the design and development of an artefact to solve the identified problem. Design science research (DSR) will be used to address the identified problem as DSR seeks to address research through the building and evaluation of artefacts designed to serve human purposes (Meyer, Helfert, Donnellan, & Kenneally, 2012). The design science paradigm is fundamentally a problem solving paradigm that seeks to create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of systems can be effectively and efficiently accomplished (Denning, 1997; Tsichritzis, 1998). A generic design science method (Figure 1.1) can be adapted to solve specific research projects by means of artefact development.

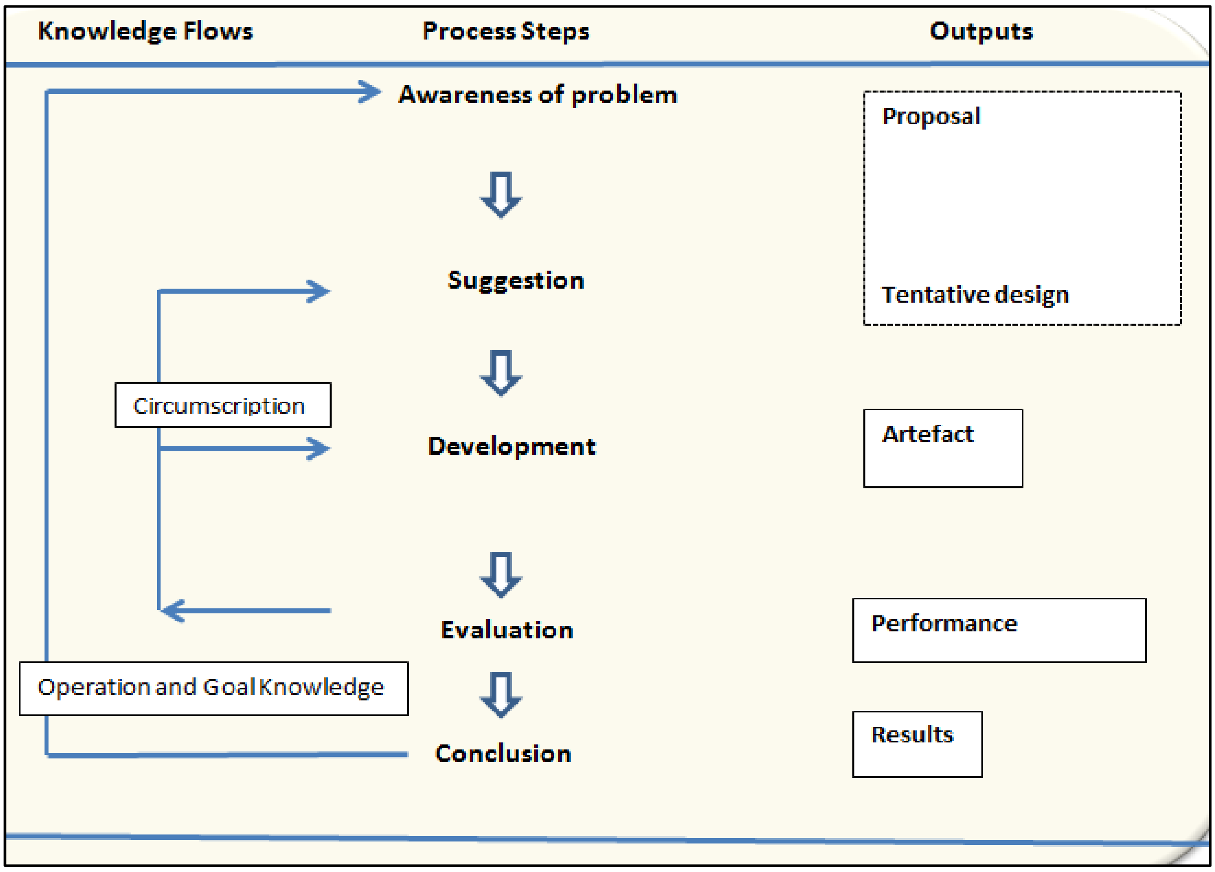


Figure 1.1: General Methodology of Design Science Research adapted from Vaishnavi & Kuechler,(2004a)

The DSR process consists of the following five phases (an in-depth discussion of each of the phases follows later in this section).

Awareness of the problem: This phase is about identification of the problem. It involves investigating and analysing a practical problem leading to the causes and identification of the problem (Johannesson, 2012). The output of this phase would be a research proposal (Vaishnavi & Kuechler, 2004a).

Suggestion: This phase involves defining requirements to plan a solution of the identified problem in the form of an artefact. New functionality based on novel outlines of existing or new solutions are envisioned and the outcome is a tentative design (Vaishnavi & Kuechler, 2004a).

Development: This phase involves the design and implementation of the suggested artefact to address the identified problems. This involves determining the functionality of the artefact, as well as the artefact's construction.

Evaluation: After the development phase, the artefact will need to be evaluated in terms of the stated requirements and objectives of the research. This evaluation will determine if the artefact realises the requirements and to what extent it solves the identified problem that the research is trying to answer (Johannesson, 2012).

Conclusion: This phase signifies the end of the research effort and the results are written up and may lead to further research if the knowledge gained so indicates.

DSR follows an iterative approach as shown by the circumscription arrows (Figure 1.1) and it is a continuous process of artefact refinement, leading to the identification of new problems and suggestions to artefact improvements.

The research questions of this research are mapped on to the DSR Methodology process model (Figure 1.2). Research questions 1, 2, 3 and 4 investigate the problem and provide solutions to the problem. These questions are mapped on to the phase related to the awareness of the problem. Research questions 5 and 6 investigate different architectures including the design of the generic NUI architecture and implementation and evaluation of the NUI architecture. These questions are mapped on to the design and evaluation phase and the outcome is the NUI architecture. The evaluation of the architecture consists of a framework implementation and incorporation of an instance of an FMS. Research question 7 is the evaluation of the implemented framework instance of an FMS. This is mapped onto the evaluation phase and the outcome of the instance evaluation determines how effective the architecture is in detecting an FMS. The final phase is the conclusion of the research and the final finding of the research.

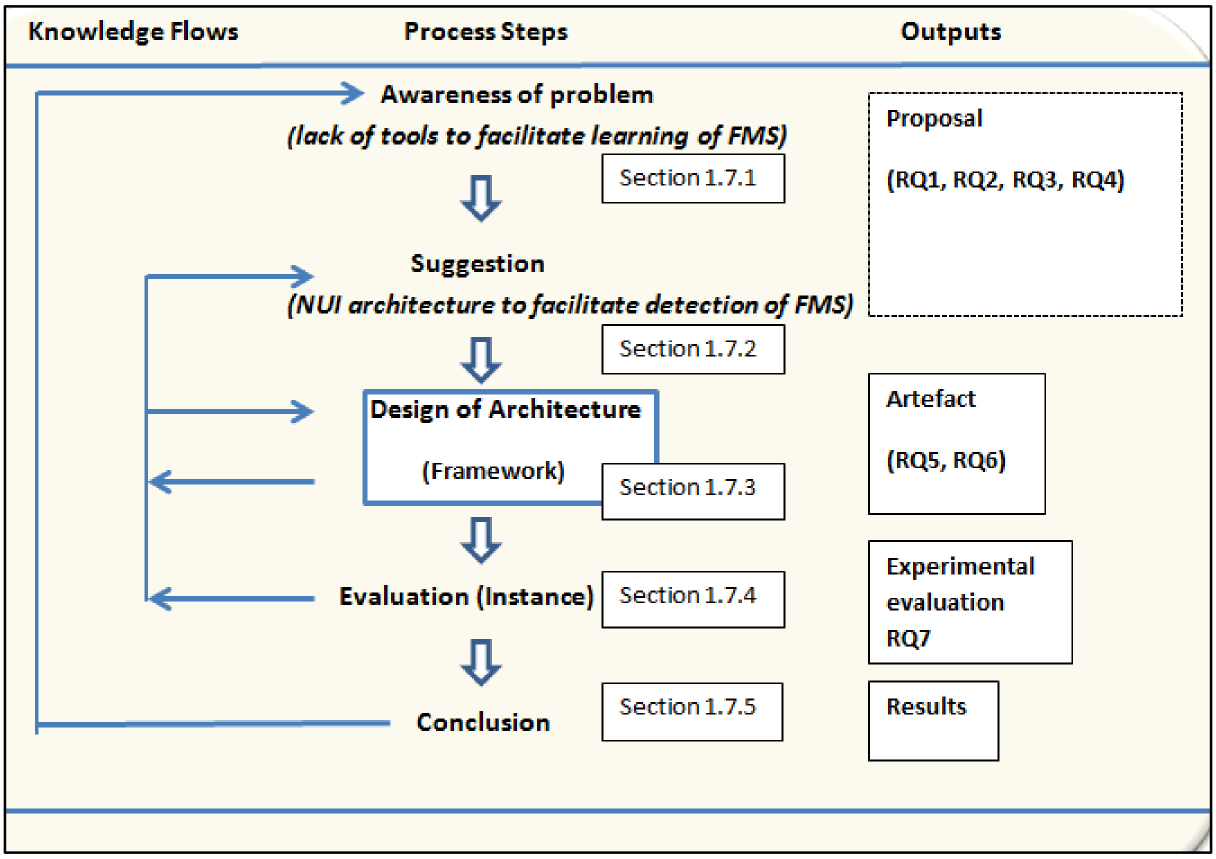


Figure 1.2: DSR Methodology process model adapted from Vaishnavi and Kuechler (2004b)

1.7.1 Awareness of the problem

The main aim of this research is to investigate how an NUI architecture can be designed and used to facilitate the detection of FMSs (Section 1.5). The increase in obesity and reduced physical activity among children has brought awareness of the problem that children may have undeveloped FMSs. The lack of mastery in FMSs in children is partly due to many teachers not being sufficiently skilled in teaching these skills (Lubans et al., 2010; Matin & Hands, 2003). Time constraints also prevent children from learning these skills as they do not have the time to practise and learn. Thus, tools can be used to encourage children to practise the correct performance of FMSs.

In order to achieve the aim of the research, an understanding of FMSs is required (RQ1). Furthermore, an investigation into the kind of challenges faced in teaching and learning of FMSs in children (RQ2) needs to be carried out. A literature study will be conducted in order to provide a critical summary of FMSs and the challenges of teaching and learning of these skills. The literature study will also include an investigation into NUIs and provide an understanding of what NUIs are. A brief comparison of motion sensing devices for Kinect, Nintendo Wii and Sony Move controller will also be conducted. The comparison of the three devices will discuss how their NUI can be used to identify the motion of the user. An investigation into the features of the NUI supported by Kinect will also be carried out including a discussion of how the Kinect is used generally (RQ3, RQ4). From this investigation, it will be shown as to why the NUI for Kinect was chosen as a suggested solution to facilitate the detection of FMSs.

1.7.2 NUI architecture to detect FMSs using the Kinect

There are many entertainment environments that allow the user to perform tasks that require bodily interaction with sensor-equipped environments. These environments may require a user to navigate a virtual building or play a game using equipment such as a head-mounted display or an exercise bicycle (Nijholt, Pasch, Dijk, Reidsma, & Heylen, 2011). Other types of interfaces that allow physical exercise like Nintendo's Dance Dance Revolution allow users to be active while playing games.

With these new interfaces that allow for full body movement, a suggestion to solving the problem identified is to develop an NUI architecture supported by Kinect that would allow the detection of FMSs (RQ3). Children love to move and an NUI supported by Kinect would allow them to be fully engaged in the activity being performed. An NUI supported by Kinect could allow children to learn FMSs by performing movements like jumping correctly in order to improve their level of a particular movement skill. By providing feedback, children could be able to learn how to perform FMSs correctly.

1.7.3 Development of an NUI architecture

One of the main activities in DSR is the design and development of an artefact. In order to realise the aim of this research (Section 1.3), an NUI architecture will have to be designed, and then applied by developing a framework and an instance to facilitate the detection of FMSs. The design of the architecture will have to adhere to software architecture best practices by following a set of guidelines defined by architectural styles. The functional requirements for the architecture to be designed will need to be defined. Based on these requirements, the most appropriate architectural style will be identified.

1.7.4 Evaluation

The design and implementation of a framework for an NUI supported by Kinect will demonstrate feasibility of the proposed architecture (RQ6). The incorporation of an instance will be the evaluation of the framework to determine its feasibility. This phase of the design science process involves iterative prototyping where the developed architecture is evaluated until it meets the desired outcome of the suggested solution. The evaluation of the instance will determine its effectiveness in detecting an FMS. An experimental evaluation will be conducted to assess how effective the instance (prototype) is in facilitating the detection of an FMS (RQ7). The NUI interface for Kinect will have to determine if the user performs an FMS correctly or not.

1.7.5 Conclusion

The conclusion discusses the outcome of the research undertaken and discusses whether the aim of the research was reached. The conclusion also discusses the finding from the research objectives and the knowledge gained from the investigation of the research objectives. The results of the evaluation will determine whether the NUI supported by Kinect is an effective tool for detecting FMSs. The evaluation will determine if the implemented architecture can be used as a template for future FMSs' inclusion and be used for detecting movement skills. The limitations and challenges faced during the course of the research with respect to determining a suitable solution to facilitate the detection of FMSs will be discussed.

1.8 Dissertation Outline

The dissertation outline (Figure 1.3) shows how the chapter contributions of this research relate to each other and how they are adapted to the chosen research methodology.

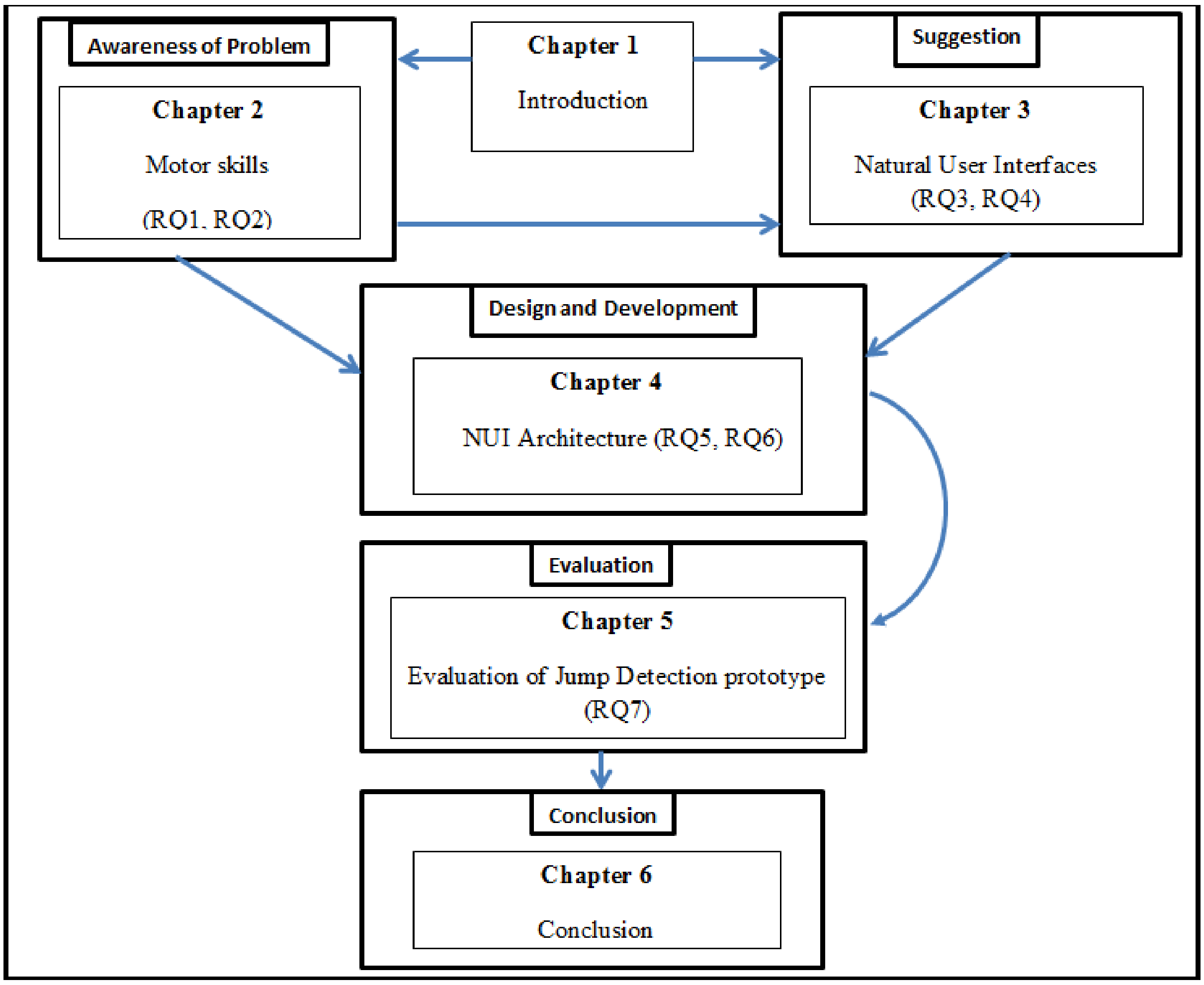


Figure 1.3: Dissertation structure

Chapter 2 will review the literature of FMSs, giving details of what they are and describing the challenges of teaching and learning these skills (RQ1, RQ2). A brief discussion of the phases of motor development will be presented to contextualize FMSs and show their importance in the development of movement skills. The literature will also review different types of FMSs that need to be learnt showing the correct criteria for the performance of each movement skill. The literature review will show how children learn FMSs and how best to teach FMSs to children. This will include a discussion of the equipment used and the types of games they play to improve their proficiency in FMSs. A review of the existing tools used to learn FMSs will be presented.

Chapter 3 (RQ3, RQ4) will discuss NUI as this leads to the suggestion of a solution to the problem statement. The discussion of NUI investigates technological supporting tools that can be used to help facilitate the detection of FMSs. A brief discussion of the history of user interfaces will give an overview of the development of NUIs. A comparison of gestural interaction devices will be conducted and this will show why the Kinect was considered the best option for this research. This will lead to a discussion on the features of the NUI for Kinect and how the Kinect detects movements. A discussion of different gesture recognition approaches for detection of movement will be presented. The discussion will give the advantages and disadvantages of each approach. From the discussion, the most appropriate gesture detection approach for the development of the selected FMS will be used for implementation. The selected FMS to be implemented will be determined by the constraints of the Kinect and the selected gesture detection approach to be used for implementation.

Chapter 4 (RQ5, RQ6) discusses the architecture for detecting FMSs using the NUI for Kinect. This chapter will present a generic design of the NUI architecture to be used for implementation. A brief discussion of software architectures will be presented, providing different guidelines to the design of architectures. The guidelines will be used to motivate the design of the NUI architecture. The designed architecture will be evaluated by implementing a framework to detect FMSs using NUI supported by Kinect. The implemented framework will determine the feasibility of the designed NUI architecture. The architecture will also be evaluated by extending the framework into an instance for a specific FMS. This will determine if the architecture can be used as a template to facilitate the development of an application to include future FMSs.

Chapter 5 (RQ7) will discuss the evaluation of the prototype. The prototype is an instance for the jump FMS incorporated into the implemented framework of the proposed architecture. The evaluation will be conducted to determine how effective the prototype is in determining the correct and incorrect performance of an FMS. A detailed discussion of the evaluation will be presented in this chapter.

Chapter 6 will be the conclusion of the research topic. This chapter will give a brief overview of the research. The limitations and contributions of this research will be discussed including future research.

Chapter 2: Motor Skills

2.1 Introduction

Movement is a facilitator that allows children to interact with their environment and with their peers. Movement in early childhood primarily focuses on the performance of motor skills such as catching, jumping, and running. Motor development (Section 2.2) in children is a gradual process of learning until they are fully able to perform motor skills proficiently in team sports or any advanced form of physical activity. Motor learning (Section 2.3) is important for children and instructors to understand how to perform movement skills. To be able to perform more advanced forms of physical activity or sports, children need to learn how to perform the fundamentals of a skill or skills (Section 2.4). It is therefore important that motor skills are taught in early childhood (Section 2.5) to allow them to participate in more advanced sports and physical activity as they become older. The problem statement (Section 1.2) identified the need for teachers to be supported with tools to teach fundamental movement skills (FMSs), because many teachers do not have the time and are not suitably qualified to teach FMS (Section 2.6).

Research Questions 1 and 2 are addressed, by presenting the role of FMSs in the proper development of children's movement skills, and discussing the challenges faced when teaching FMSs.

2.2 Motor Skill Development

Motor skill development is a process that involves the improvement of movement abilities. This process of motor development involves the development of a child's bones, muscles and the ability for the child to move about and manipulate his or her environment (PDCE Admin, 2010). The process of improvement of motor skills is not dependent on age, and development can differ among individuals of the same age (Haywood & Getchell, 2009). The development of motor skills follows a sequential process and more complex motor skills follow the more basic and simple movement patterns. Motor development is a lifelong process beginning with an infant being able to move its head and trunk, continuing all the way to adulthood when the fully grown adult is involved in athletics and sports. The development of motor skills is achieved through learning and adaptation to the environment from childhood to adolescence (Clark, 2007).

A motor skill refers to a task which allows the performer to accomplish a specific act using the precise movement of the muscles (EduCLIME, 2013). The tasks to be performed involve the use of gross motor skills or fine motor skills. Gross motor skills require the use of large muscle groups to perform tasks like walking, running and jumping. Fine motor skills involve the use of smaller muscles to perform tasks that need accuracy and precision such as writing and playing the piano. Gross motor skills are usually referred to as FMSs as they form the building blocks of more complex physical movements (LeapFrog, 2013).

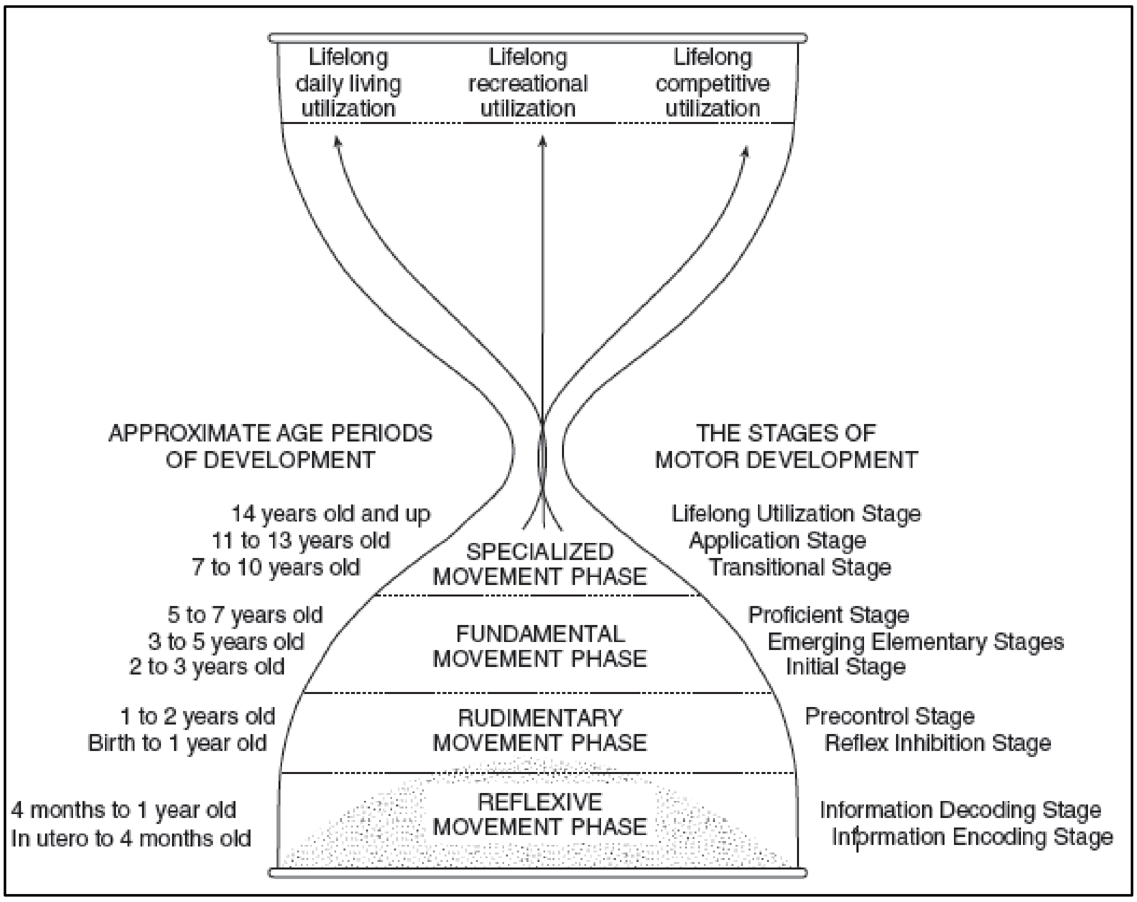

Lifelong daily living utilization
Lifelong recreational utilization
Lifelong competitive utilization
APPROXIMATE AGE PERIODS OF DEVELOPMENT
THE STAGES OF MOTOR DEVELOPMENT
14 years old and up
11 to 13 years old
7 to 10 years old
SPECIALIZED MOVEMENT PHASE
Lifelong Utilization Stage
Application Stage
Transitional Stage
5 to 7 years old
3 to 5 years old
2 to 3 years old
FUNDAMENTAL MOVEMENT PHASE
Proficient Stage
Emerging Elementary Stages
Initial Stage
1 to 2 years old
Birth to 1 year old
RUDIMENTARY MOVEMENT PHASE
Precontrol Stage
Reflex Inhibition Stage
4 months to 1 year old
In utero to 4 months old
REFLEXIVE MOVEMENT PHASE
Information Decoding Stage
Information Encoding Stage

Figure 2.1: The phases of Motor Development (Gallahue, Ozmun, & Goodway, 2012)

The process of motor development of a child is attained through movement patterns and skills (Malina, 2003). The reflexive movement, rudimentary movement, fundamental movement, and specialised movement phases (Figure 2.1) constitute the four phases of motor development. These phases of motor development represent the sequential pattern that a child follows during its period of motor development. The child’s motor development progresses from very simple reflexive movements as a newly born baby to more specialised movements in adolescence. The phases of motor development indicate the stage in which most children are during their development of motor skills. It is important to remember that most children pass through the four phases at different times. The phases of motor development serve as an indicator of a child’s progress when teachers observe children’s motor skill development (Krog, 2010). This knowledge obtained from the child’s motor development gives teachers the signs as to when and how to teach children and what type of movements an individual child needs to develop (Krog, 2010).

Reflexive movements are the first movements that a foetus makes (Gallahue et al., 2012). Reflexes are involuntary or automatic controlled movements that form the foundation for the phases of motor development. The reflexive movements consist of primitive and postural reflexes. Primitive reflexes are the survival mechanisms for a new born baby and postural reflexes help control body movements of the baby (Krog, 2010).

Rudimentary movements are the first forms of voluntary movements performed by a child (Krog, 2010). They are usually observed from birth to about the age of two. The rudimentary movements of a child denote the basic forms of maturation reliance on voluntary movement vital for survival (Gallahue et al., 2012). These movements involve grasping, reaching, and releasing as well as stability movements such as gaining control of the head, neck and trunk muscles. Rudimentary movements form the basis for fundamental movement development.

The **fundamental movement** phase happens between the ages of two and seven as the child gains control over his or her gross motor skills (Atchley et al., 2012). These movement skills begin to develop when the child is able to walk and move freely (Gallahue et al., 2012). The child learns the basic loco-motor, non-loco-motor (Section 2.4.1) and manipulative (Section 2.4.2) skills during this phase. Fundamental movements form the building blocks for more specialised movements (Okely et al., 2001).

The **specialized movement** phase is a period when movement is used as a tool to be applied to a variety of more advanced movement activities for recreation, sport and everyday living (Gallahue et al., 2012). The specialised movement phase begins at the age of eight and continues into adulthood. Children who have mastered the previous three phases are more likely to enjoy more complicated physical activities and engage in competitive team sports and games. FMSs like hopping and jumping, can be applied to rope-jumping activities, performing dance moves or triple jump in athletics (Gallahue et al., 2012).

The focus of this research will be on the third phase of motor development which is the fundamental movement phase. The process of learning FMSs follows a series of stages that a child goes through from being a beginner, to becoming proficient in the performance of a movement skill. Understanding the process of motor learning, gives teachers some ideas of how to help children learn FMSs. Teachers can assess the correct performance of an FMS by observing the attempts made by the child.

2.3 Motor Learning

Motor learning is the process that allows for the improvement of motor skills through practice. It involves the acquisition of motor skills as well as enhancing the performance of learned or more experienced motors skills. The re-acquisition of skills that are considered difficult to perform or cannot be performed due to injury or disease can also be improved through motor learning (Millslagle, 2003). Motor learning requires learners to acquire an idea of movements or understand the basic pattern of coordination (Gentile, 1972).

When a child begins to learn movement skills, he/she would need to acquire and gather information about the skill to be learnt. This information can be gathered from verbal or visual instruction on how the skill is performed and the child can then try to perform the skill. This allows the child to put different actions of the skill together and with repeated practice, instruction and adequate amount of time, he/she can become proficient in performing the skill.

The stages of motor learning involve the cognitive stage which is the information gathering stage; the associative stage when the child begins to perform a skill; and the autonomous stage when the child becomes proficient in the performance of a skill. FMSs are learnt in the cognitive and associative stages. Proficiency of an FMS is achieved in the autonomous stage of motor development.

2.3.1 Cognitive Stage

The cognitive stage forms the basis of the development stage of motor learning. This stage begins with the learner being introduced to a motor task. It solves the problem faced by a new learner on understanding what to do (Schmidt & Lee, 2005). Learning a new skill would be difficult for a learner if he/she has not received prior knowledge about the skill, the knowledge being either visual or verbal. The cognitive stage signifies the beginning of motor learning and the processing of information (Huber, 2012). During this stage, the learner must determine the objective of the skill, as well as understand the relational and environmental cues that control and regulate the movement.

The cognitive stage is focused on the movement or performance of the skill by the learner. During this stage the learner benefits from instruction, guidance and feedback to improve performance of the skill. It is important, therefore, that the learner is given the required information, assistance, and time to develop good basics of movement (Huber, 2012).

2.3.2 Associative Stage

The associative stage is characterised by having much less verbal information, and more of smaller gains in performance, conscious performance, adjustment making, awkward movements and taking a long time to complete a task (Huber, 2012). During this stage, the learner is more concerned with the performance and refinement of the skill. This allows the learner to focus on the task and improving his/her performance of the task (Schmidt & Lee, 2005). This stage is also known as the motor stage as the main aim of the problem to solve is learning how to perform the skill (Adams, 1971). The learner begins to make fewer gross errors, and should be provided with instructional feedback for error correction in order for him or her to quickly improve and master the skill.

2.3.3 Autonomous Stage

The autonomous stage is the highest level of proficiency in motor learning. It is characterised by an almost automatic kind of performance. During this stage, the learner shows consistency and confidence in doing the task and few errors are observed. It would require many years of practice to arrive at this stage; therefore many learners would not be able to reach this stage (Huber, 2012). At this stage, cognitive processing demands are minimal and the learner focuses on processing other information, such as form or style of movement in sports like dancing, soccer and swimming (Schmidt & Lee, 2005). But just because movement is performed automatically during this stage, this does not mean that the movement is correct or can be maintained. Therefore, it is important that learners understand the movement skill during the cognitive stage and how to perform the skill correctly during the associative stage

(Huber, 2012). If automatic movements are done incorrectly, the learner will have to relearn the skill by going back to the cognitive and associative stages.

2.4 Fundamental Movement Skills

FMSs are movements that have specific observable patterns (Gallahue et al., 2012). By making use of the different body parts, these movement skills are described as the building blocks of many physical activities. They are known as the ABCs of athleticism (agility, balance, coordination and speed) and they include locomotor skills, non-locomotor skills (Section 2.4.1) and manipulative skills (Section 2.4.2). Locomotor skills (running, jumping, hopping, galloping, skipping, and leaping) are concerned with the movement of the body from one place to another, while non-locomotor movements (balance, twisting, turning and dodging) are more static. Manipulative skills (throwing, catching, kicking, and striking) are more concerned with handling objects using hands and feet. FMSs are easily developed when children are young and this enables them to learn more advanced forms of movements. The development of FMSs in children is a major contributor to the lifestyle children will adopt in their adolescent years and into adulthood. Children who do not acquire these skills early on will have difficulties acquiring more complicated movement skills and this could lead to ridicule from peers and avoidance on their part to participant in team sports.

2.4.1 Locomotor and Non-locomotor skills

Locomotor skills refer to the movement of the body from one place to another (Portela, 2007). The refinement of these skills leads to the application of the skills to specific sports such as high jump in track and field (Gallahue & Donnelly, 2003). Non-locomotor skills, also referred to as stability skills, do not involve movement of the body from one place to another but are performed in a stationary position, that is, while standing, sitting or lying on the ground. Non-locomotor skills are not as well defined as locomotor skills (Portela, 2007). Since all the movement in locomotor and manipulative skills involve stability, non-locomotor skills usually form the basis for these movement skills. Dodging an opponent in a sport would be considered a non-locomotor skill because much emphasis is placed on maintaining equilibrium throughout the task. Locomotor and non-locomotor skills need to be developed because they form the foundation of coordination, balance, strength and stamina. Environmental factors such as opportunities for practice, encouragement and instructions, play an important role in the acquisition of advanced patterns of locomotor movement (Gallahue & Donnelly, 2003).

The following sections will give a detailed discussion of the individual locomotor and non-locomotor skills. The discussion will give a description of each of the FMSs, their importance in sports and the skill component involved in the performance of the skill. This information will lead to the identification of specific movement patterns to be observed for each of the movement skills. The locomotor skills to be discussed include running, leaping, jumping, hopping, galloping and skipping. The non-locomotor skills to be discussed include balance and dodging.

Running

Running (Figure 2.2) is characterized by the rapid movement of the feet. For a movement to be considered a run, the gait should constitute a flight phase where both feet are off the ground. The performance of running in a child can be observed by making sure that the sub-skills are performed correctly. The following points highlight the specific observable patterns of running:

- Eyes should be focused forward throughout the run
- Knees should bend at right angles during the recovery phase
- Arms should bend at elbows and move in opposition to legs
- Contact the ground with front part of foot
- Body should lean slightly forward.

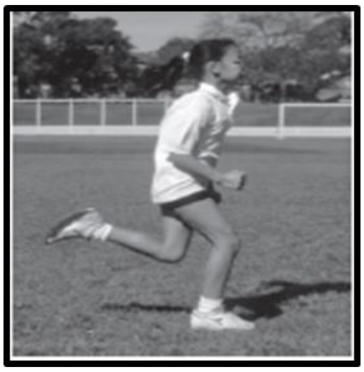


Figure 2.2: Demonstration of running (NSW Department of Education and Training, 2000)

Children love running and when given a chance, they can be seen running and leaping from place to place. The patterns of running and leaping occur early in childhood. Children attempt to run when they reach the age of eighteen months but this attempt just resembles walking fast because there is no flight phase (Haywood & Getchell, 2009). The learning of this skill is fundamental to many games, sports and everyday activities. With the proper encouragement, practice and instruction, the running pattern should continue to improve and children should reach the proficiency stage by age seven. Running is used in sports such as athletics, soccer and basketball and can also be used in everyday situations like running for a bus. Being proficient in the running technique improves speed and endurance, which can also enhance health-related fitness by improving cardiorespiratory endurance.

Leaping

Leaping (Figure 2.3) is a locomotor skill that can be viewed as an extension of the running pattern. The leap is distinguished by a longer flight phase when performing the task. It is usually performed in a single phase, unlike the run which is a repeated skill when performed (Gallahue & Donnelly, 2003). The observable sub-skills of leaping are listed below:

- Forward movement should be sustained throughout the leap
- Eyes should be focused forward throughout the leap
- Take off from one foot and land on the opposite foot
- During flight legs are straightened with the arms held in opposition to legs
- There should be controlled landing without losing balance.

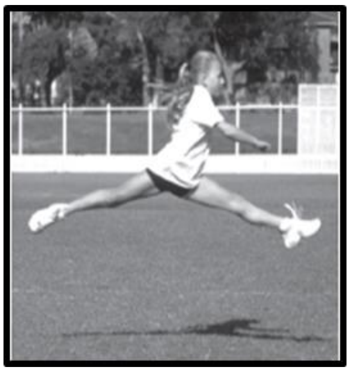


Figure 2.3: Demonstration of leaping (NSW Department of Education and Training, 2000)

The leap can be performed in a stationary position or can be followed after a run. Leaping is used in play and recreational activities such as hopscotch and crossing a stream. It is also used in professional sports and athletics like leaping over a hurdle in track and field. Since the patterns of running and leaping are important to everyday activities, it is vital that they are developed to a more mature level.

Jumping

The FMS of jumping (Figure 2.4) is used in a range of sports, as well as fun and daily activities. Children first attempt jumping tasks at a young age, doing simple forms of jumping before the age of two (Haywood & Getchell, 2009). Jumping is an example of a locomotor skill that involves individuals propelling their bodies from a surface and being able to jump as high as possible with either one or both feet off the ground (Department of Education, 2013). The observable sub-skills of jumping are listed below:

- Eyes should be focused forwards or upwards throughout the jump
- Crouch with your knees bent and arms behind body
- There should be forceful upward thrust of arms as legs straighten to take off
- Contact the ground with front part of feet and bend knees to absorb force of landing
- Landing should be balanced with no more than one step in any direction.

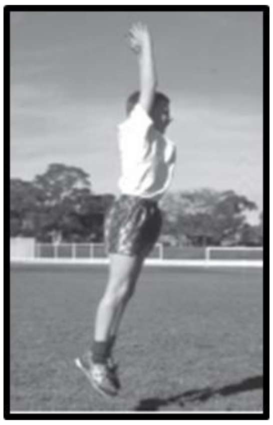


Figure 2.4: Demonstration of jumping (NSW Department of Education and Training, 2000)

Jumping can be performed on a horizontal plane, vertical plane or from a height. Horizontal jumping is often performed with the intention of jumping for distance using both feet. Practice of this skill can be done on the gymnasium floor, over low obstacles or out in the fields. Vertical jumping involves taking off the ground with one or both feet in an upward direction and landing on both feet. This skill is used as a measure of strength, skill performance and coordination of movement. Jumping from a height involves jumping from a high station with both feet and landing safely on the ground. Both feet should be able to absorb the force of the jump in a manner equal with the height of the jump (Gallahue & Donnelly, 2003). Proficiency in the jumping skill is fundamental for sports such as gymnastics, basketball, volleyball and rugby.

Hopping

Hopping (Figure 2.5) is a specialised form of jumping. It is an example of a locomotor skill that is characterised by jumping with one foot and landing on the same foot, often repeatedly (National Ireland Curriculum, 2013). The observable sub-skills of hopping are listed below.

- Foot on non-support leg is bent and carried behind the body
- Landing and springing should be from the ball of the foot
- Hold the head still with eyes looking forward
- Rhythmical movement should be in a straight line
- There should be controlled landing without losing balance.

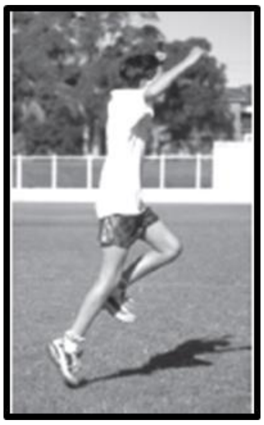


Figure 2.5: Demonstration of Hopping (NSW Department of Education and Training, 2000)

Hopping usually involves rhythmical loss and regain of support in an upward, forward, sideways or backward direction on the same foot. It is used in athletics in the triple jump as well as on the playground with games like hopscotch. Proficiency of this skill is a good indicator of being able to maintain balance while moving.

Sliding and Galloping

The FMSs of sliding and galloping (Figure 2.6) involve the essential movements of stepping, hopping and leaping. Galloping and sliding consist of the movement of one step using one foot, then a leap-step on the other foot. Sliding and galloping are similar in many ways but differ in the direction of movement of the performed task (Haywood & Getchell, 2009). The observable sub-skills of side galloping are listed below:

- Step to the side and draw the other foot up quickly to the first foot
- Repeat the action, landing with the same foot
- Use your arms only as needed for balance
- Move on the balls of your feet
- Keep your knees bent slightly
- Lean slightly forward at the waist.

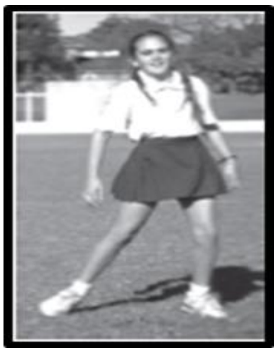


Figure 2.6: Demonstration of Galloping (NSW Department of Education and Training, 2000)

In galloping, the movement of the step can be in the forward direction or sideways in side galloping. Sliding involves the movement of the step in a sideways direction. Before children can explore the skill of sliding, it is generally more appropriate for them to experience success with the skill of galloping first (Gallahue & Donnelly, 2003). Sliding is used in a variety of sports, games, dancing and gymnasium activities that require rapid sideways movements.

Skipping

Skipping (Figure 2.7) is a locomotor skill that is distinguished by a step and hop on the same foot, with alternating feet during the performance of the skill. The observable sub-skills of skipping are listed below:

- Eyes should be focused forward throughout the skip
- Repeat the action, landing with the same foot
- Use your arms only as needed for balance
- Move on the balls of your feet
- Keep your knees bent slightly
- Lean slightly forward at the waist.

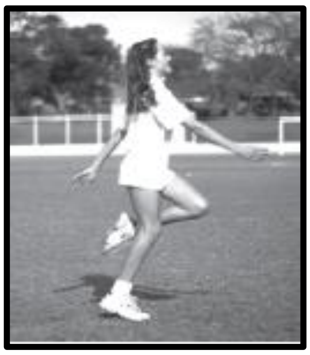


Figure 2.7: Demonstration of Skipping (NSW Department of Education and Training, 2000)

Dynamic and Static Balance

Balance (Figure 2.8) has been defined as the ability to maintain one's equilibrium in relation to the force of gravity, be it in a static position or during a performance of a dynamic activity (Gallahue & Donnelly, 2003). The observable sub-skills of Static Balance are listed below:

- Support leg should be still and foot flat on the ground
- Non-support leg should be bent and not touching the support leg
- Head should be stable and eyes focused forward
- Trunk should be stable and upright
- There should be no excessive arm movements.

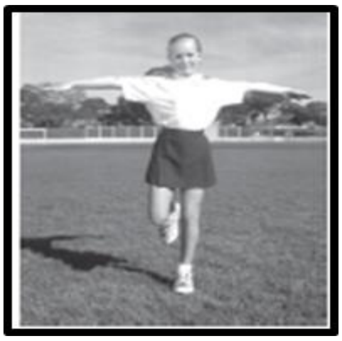


Figure 2.8: Demonstration of Static Balance (NSW Department of Education and Training, 2000)

Balance as an example of a non-locomotor skill, is an essential necessity of almost all movement skills (Portela, 2007). Static balance is defined as being able to maintain a stationary position throughout the movement (National Ireland Curriculum, 2013). Standing in one place, balancing on a board, and being able to stand on one foot are examples of static balance. The static balance on one foot is an important non-locomotor skill that is used in gymnastics, dance, diving and several team based sports. Being able to perform the static balance for a specific period of time reduces the risk of falling and physical injury. Dynamic balance happens as an individual moves through space and his/her body is controlled and stabilised through the task. During the performance of a dynamic balance activity the centre of gravity is always changing. Balance is usually the basis from which all other controlled movements begin. For children to have total control of their whole bodies, they need to have developed and become proficient in balance skills (Gallahue & Donnelly, 2003).

Dodging

Dodging (Figure 2.9) is an FMS that involves the locomotor skill of sliding and the non-loco-motor skill of balance. It involves quick, deceptive changes in direction to elude, chase or escape from an opponent (Sport New Zealand, 2012a). The observable sub-skills of Dodging are listed below:

- Change direction by bending the knee and pushing off the outside foot
- Change of direction occurs in one step
- Body should be lowered during change of direction or in the direction of travel
- Eyes should be focused forward
- Dodge should be repeated equally well on both sides.

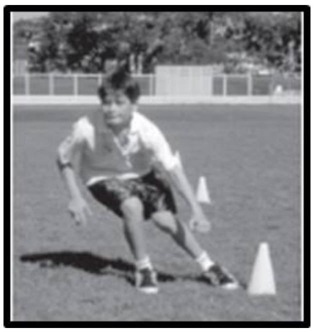


Figure 2.9: Demonstration of Dodging (NSW Department of Education and Training, 2000)

To be able to perform the dodge, the knees need to be bent and the body should shift rapidly in a sideways direction. Dodging is a common skill performed in sports, games and play activities of children (Gallahue & Donnelly, 2003). The skill is characterised by the movement of the shoulders, head, eyes or other body parts to mislead or fake the opposition.

This is seen in sports like basketball when getting around an opponent to score a basket or a tag game when avoiding being tagged.

2.4.2 Manipulative skills

Manipulative skills are movement skills that involve the movement of objects using hands and/or feet (Portela, 2007). They are gross body movements in which the performer imparts force to objects or absorbs force from received objects. The acquisition of these skills allows for the purposeful and controlled interaction with objects in the environment (Gallahue & Donnelly, 2003). These skills include throwing, catching, kicking and striking. Proficiency in these skills forms the foundation for more advanced sports like football, rugby, tennis and javelin throw in athletics. The following subsections discuss the FMS of throwing, catching, kicking and object striking.

Throwing

The overarm throw (Figure 2.10) is an example of a manipulative skill that is frequently used in many sports, such as cricket, softball and baseball. The observable sub-skills of throwing are listed below:

- Eyes should be focused on target area throughout the throw
- Stand side-on to the target area
- Throwing arm moves in a downward and backward arc
- Step towards target area with foot opposite the throwing arm
- Hips then shoulders should rotate forward
- Throwing arm follows through, down and across the body.

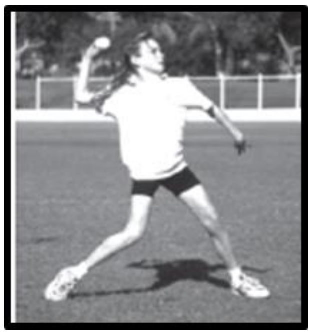


Figure 2.10: Demonstration of overarm throw (NSW Department of Education and Training, 2000)

Throwing is performed by imparting force on an object by making use of the hands. The action of throwing takes many forms such as the overarm, underarm or sidearm pattern depending on the purpose of the throw. The overarm throw is the most commonly used skill in sports and in play and games (Gallahue & Donnelly, 2003). This action is also used in athletics with the javelin, and the overhead serve smash in tennis.

Catching

Catching (Figure 2.11) is an example of a manipulative skill that involves being able to absorb and control the force of an object using the hands. The observable sub-skills of catching are listed below:

- Eyes should be focused on the object throughout the catch
- Feet should move to place the body in line with the object
- Hands should move to meet the object
- Hands and fingers should be relaxed and slightly cupped to catch the object
- Catch and control the object with hands only
- Elbows bend to absorb the force of the object.

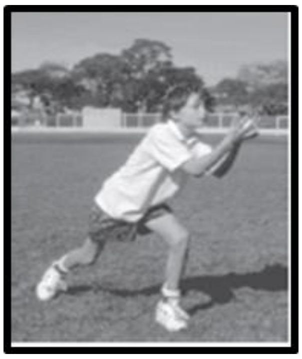


Figure 2.11: Demonstration of catching (NSW Department of Education and Training, 2000)

The main purpose of catching is to retain possession of the object you catch. The receiver should be able to catch an object in the hands rather than trap it against the body or opposite arm (Haywood & Getchell, 2009). Proficiency in the catching skill is important in most sports and games that involve objects such as balls. Examples of these sports are cricket, netball and basketball.

Kicking

The kick (Figure 2.12) is an example of a manipulative striking skill that is categorised by producing force from the foot to an object (National Ireland Curriculum, 2013). The observable sub-skills of kicking are listed below:

- Eyes should be focused on the ball throughout the kick
- There should be forward and sideward swing of arm opposite kicking leg
- Non-kicking foot should be placed beside the ball
- Bend knee of kicking leg at least 90 degrees during the back-swing
- Contact ball with top of the foot or instep
- Kicking leg follows through high towards target area.

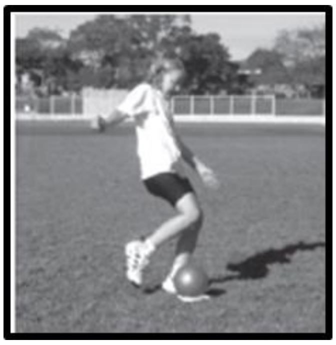


Figure 2.12: Demonstration of Kicking (NSW Department of Education and Training, 2000)

This skill can include striking a stationary or moving object with feet and may take the form of kicking a pebble, a can, or ball (Sport New Zealand, 2012b). The focus of this skill is the standing place kick and involves kicking an object which is motionless. Kicking is the basic skill in all football sports.

Object Striking

Object striking (Figure 2.13) is a manipulative striking skill that can be performed in several different planes with the use of an implement. The performance of this skill may involve contact with a moving or stationary object (Gallahue & Donnelly, 2003). The observable sub-skills of Object Striking are listed below:

- Stand side-on to target area
- Eyes should be focused on the ball throughout the strike
- Hands should be next to each other with bottom hand matching the front foot

- Step towards target area with front foot
- Hips then shoulders should rotate forward
- Ball contact should be made on front foot with straight arms
- Follow through with bat around the body.

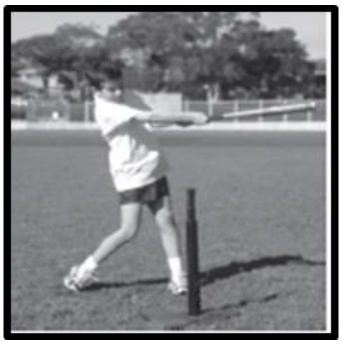


Figure 2.13: Demonstration of Object Striking (NSW Department of Education and Training, 2000)

Object striking makes use of the stability skill of balance, and is achieved when the feet and legs are comfortably spread to provide a wide stable base of support. The stability skill of balance is important because it provides a secure base of support for the strike (Sport New Zealand, 2012c). Object striking takes many forms and the skill application to sport varies. For example, the horizontal striking pattern is used in baseball while the vertical striking pattern is used in sports such as tennis, golf, volleyball and handball.

2.4.3 Common observable sub-skills of different body parts

Understanding the different types of FMSs allows teachers and instructors to better plan how to teach these skills. The description of each FMS shows the importance of learning the correct performance of an FMS and how each contributes to the child's integration into team sports, games and physical activities. Each of the different movements has specific observable sub-skills that can guide teachers and instructors in teaching FMSs. By observing these sub-skills when the child is performing a skill, teachers can give correct feedback on performance and how the child can improve the next time. The observable sub-skills focus on the movement of the different parts of the body. The observable sub-skills can be categorised and grouped into different body parts, namely the head, arms, knees and feet. The common sub-skills that correspond to each of the different individual body parts are grouped together. The following tables show a summary of all sub-skills grouped in categories for the head (Table 2.1), arms (Table 2.2), knees (Table 2.3) and feet (Table 2.4).

Table 2.1: Observable sub-skills for the head

	Locomotor						**Non-loco-motor**		**Manipulative**			
Skills	running	leaping	jumping	hopping	gallopin	skipping	balance	dodging	throwing	catching	kicking	object striking
Eyes should be focused forward	✓	✓	✓	✓		✓	✓	✓	✓	✓		
Hold the head still				✓		✓	✓					
Eyes should be focused on target area									✓	✓	✓	✓

Table 2.2: Observable sub-skills for the arms

	Loco-motor						**Non-loco-motor**		**Manipulative**			
Skills	running	leaping	jumping	hopping	galloping	skipping	balance	dodging	throwing	catching	kicking	object striking
Forceful upward thrust of arms			✓									
Use your arms only as needed for balance					✓		✓					
Throwing arm moves in a downward and backward arc									✓			
Throwing arm follows through, down and across the body									✓			
Hands move to meet the object										✓		
Catches and controls the object with hands only										✓		
Follows through with bat around the body												✓
arms behind body			✓									

Table 2.3: Observable sub-skills for the knees

	Locomotor						**Non-loco-motor**		**Manipulative**			
kills	running	leaping	jumpin	hopping	gallopin	skippin	balance	dodging	throwin	catchin	kicking	object striking
Knees should bend at right angles during the recovery phase	✓											
Crouch with your knees bent and			✓									
Bend knees to absorb landing force			✓		✓	✓						
Non-support leg should be bent and not touching the support leg							✓					
Change direction by bending the knee and push off the outside foot								✓				
Bend knee of kicking leg at least 90 degrees during the back-swing											✓	
Straighten legs to take off			✓									

Table 2.4: Observable sub-skills for the feet

	Locomotor						**Non-loco-motor**		**Manipulative**			
Skills	runnin	leapin	jumpin	hoppin	gallopi	skippi	balanc	dodgin	throwi	catchin	kickin	object strikin
Contact the ground with front part of foot	✓		✓	✓	✓	✓						
Take off from one foot and land on the opposite foot		✓										
Landing must be balanced with no more than one step in any direction			✓									
Foot on non-support leg is bent and carried behind the body				✓			✓					
Spring from the ball of the foot				✓								
Step to the side and draw the other foot up quickly to the first					✓							

foot												
Support leg should be still and foot flat on the ground							✓					
Step towards target area with foot opposite the throwing arm									✓			✓
Non-kicking foot must be placed beside the ball											✓	
Contact ball with top of the foot or instep											✓	
Kicking leg follows through high towards target area											✓	

The common observable sub-skills of the head stress the importance of having the eyes facing forward when performing a locomotor and non-locomotor skill, while manipulative skills all have a common component skill of the eyes focusing on the object. The importance of keeping the head still when learning an FMS is shared between locomotor and non-loco-motor skills of hopping, skipping and balance.

The arms category have unique component skills for each of the movements except for the FMS of balance and galloping which share the component of using the hands for balancing. The knee category has a shared common component of bending the knee with the locomotor skill of jumping, galloping and skipping. The rest of the sub-skills have their own unique movements of the knees. The feet category consists of a shared component skill of the feet landing on the ground for the locomotor skills of running, jumping, hopping and skipping.

2.5 Teaching and Learning Fundamental Movement Skills in Early Childhood

Children have limited FMSs and these skills do not develop automatically as they age as is commonly believed (Pica, 1997), but rather, they require explicit teaching in order to develop proficiency or mastery of these skills (Gallahue & Ozmun, 2002). Environmental settings that include opportunities for practice, encouragement, and instruction are vital to the development of FMSs (Payne & Isaacs, 2002). Therefore, in order for children to develop these skills, they must be instructed and practised.

The best learning period for development of FMSs is between the ages of two and seven (Gallahue & Cleland-Donnelly, 2007). During this period, children are highly enthusiastic and interested in learning. They have also developed physically and are capable of benefitting from instruction in physical education (Leeds Metropolitan University, 2004). It is important for teachers to understand that children develop and learn at different paces (Hardy, King, Farrell, Macniven, & Howlett, 2010), so teachers need to choose appropriate activities depending on the level of the child. The training activity includes task difficulty level and the equipment to be used. Teachers need to use visual demonstrations, and also provide instructions and encouraging feedback to children.

According to Mitchell et al. (2011), a tailored FMSs physical education class led by a teacher improves movement skills competency in all children. Teachers should be encouraged to teach movement skills using a mastery learning environment where the child is able to succeed and is encouraged to personally improve. This environment can lead to high levels of satisfaction and better levels of observed competence (Theeboom, De Knop, & Weiss, 1995). According to Booth et al. (1999), children would be able to master FMSs quickly if they were given appropriate instruction to perform a given task. The children should also be given feedback on the performance of the task coupled with sufficient practice opportunities. FMSs can be learned through participation in suitable teaching which most children would find enjoyable (Booth et al., 1999).

2.5.1 Learning through Play and Games

Children seem to define themselves and their world through play. Play provides enjoyment for the children and is also serious in that it provides opportunities for enhancing a child's feelings of mastery. It promotes the sense of being important and encourages social involvement with other children.

Unlike play, games are usually structured and have predictable outcomes (Morris & Steal, 1999).Games are activities that are described by implicit rules, in which one or more contestants engage in a cooperative, collaborative, or competitive play in order to produce a predictable outcome. They have voluntarily agreed contests with agreed upon rules and clearly defined goals (Gallahue & Donnelly, 2003). Children learn most effectively through fun and playing games because the environment is motivating and they can control and choose their own level of risk (Sport New Zealand, 2012d). Playing games gives them the means to express, explore and learn more about themselves and others. Situations involving play can be good opportunities for children to practise, experiment, consolidate and become proficient in FMSs (Department of Education, 2013). Providing children with tools that allow them means to move and experiment can be used to give children an opportunity to learn

FMSs. Teachers can make use of music to encourage children to participate in movement skills. The child learns by moving to the beat of the music when attempting FMSs.

Games and play encourage children to participate in learning FMSs; however, teachers need to get involved and make sure that children are performing correct movements. Leaving children to play without instruction provides learning by awareness only and not proficiency. It has been shown that learning FMSs using play and games only is not efficient for children to master FMSs (Department of Education, 2009). The teachers should select a particular FMS to be learnt and observe that the children perform the sub-skills required. Teachers should organise lessons that have a mixture of play and games, as well as tailored instructions of an FMS to be performed.

2.5.2 Assessment of FMSs in children

Assessment is used to measure the current level of performance of a child's movement skills. This type of assessment is usually known as formative assessment. Formative assessment is required to determine the next stage of learning that will occur and it allows the programme to fit the needs of the child (Achper Victorian Branch, 2008).

Assessment of motor skills takes two primary forms, namely, process assessment and product assessment. Process assessment is a qualitative approach that focuses on the form, style, or observed performance of an FMS. In product assessment the focus is more on the quantitative performance and is more concerned with how far, how high or how many (Gallahue & Donnelly, 2003). When a child begins to learn a new skill, learning occurs slowly and performance of the whole skill would not be an appropriate measure of learning and improvement. A more appropriate form of assessment would be a qualitative type which is more concerned with the quality of the performance of the skills rather than the outcome (Achper Victorian Branch, 2008). For example, for a beginner in tennis, it would be more appropriate to demonstrate the key components of hitting the ball with a tennis racket than how many times the ball is hit against the wall.

To teach and assess FMSs, teachers provide learners with equipment such as balls, hoops and skipping ropes. The children are instructed, for example, to kick the ball as far as they can or to jump over an obstacle. This allows the children to become familiar with movements such as jumping, kicking and other FMSs. Children can be directed on how to perform an FMS correctly by focusing on the sub-skills of the selected FMS. The teachers provide visual cues of how each skill is performed and observe the child's performance. The child's level of achievement is assessed by checking the number of sub-skills performed correctly.

There is some evidence which suggests that early childhood teachers may not be sufficiently well prepared to implement high quality movement programs (Breslin, Morton, & Rudisill, 2008). Current findings according to Hardy et al. (2010) stress the need for teachers to provide structured opportunities which enable children's development and confidence in FMSs. Booth et al. (1999) suggest that sufficient curriculum time and suitably qualified and resourced staff need to be made available to support FMS development among primary school children.

2.6 Challenges of Teaching Fundamental Movement Skills in Early Childhood

Due to the mistaken belief that children's motor skill development is automatically acquired through maturation, the need for both practice and instruction of FMSs in early childhood has been undermined (Clark, 2007). It is true that some FMSs like walking will develop naturally as the child matures, but most of the FMSs used in more advanced sports need to be taught and assessed for children to reach mastery (Olrich, 2013). Teachers often choose to allow children to engage in free play which leads to some children choosing to be inactive when they are not actively involved in the play (Krog, 2010). According to Clark (2007), teachers and parents often think that teaching movement skills to children is not necessary for children under the age of ten years old. This flawed misconception leads to less emphasis for structured motor development and intentional teaching by the teachers (Krog, 2010).

Time constraints have also forced teachers to resort to allowing children to engage themselves in games and play without instruction. Insufficient time allocated in the curriculum for the development of these skills means children are not likely to have the time to learn and practice even the most basic skills (Ennis, 2011). Since there is limited time to teach movement skills, many teachers feel that it would be beneficial for children to engage in games and at least understand the movement involved (McCaughtry, Barnard, Martin, Shen, & Kulinna, 2006). When children are seen for a limited amount of time, teaching and assessing movement becomes a concern for the teachers. Since assessment must be done during the activity, the period known as the formative period provides the best opportunity for teachers to instruct and assess movement skills (Olrich, 2013). The findings by Booth et al. (1999) suggest that a total of approximately ten hours of instruction is necessary for most children to master a particular FMS.

A further challenge faced with the teaching of FMSs is the worldwide problem of inexperienced individuals who are not qualified to teach or implement appropriate movement programmes for children (Fredericks, Kokot, & Krog, 2006; Gagen & Getchell, 2006). From workshops conducted by a South African researcher Krog (2010) between 2003 and 2009, it was found that most of the educators were not aware of the influence of poor motor development of a child and how this affects his/her overall development at school. Movement skills are being taught by generalist teachers who do not have the appropriate qualification to teach these skills. This leads to disorganized lessons or supervised play to teach children movement skills and the focus is not on the development of the skills. The generalist teachers do not have the confidence to teach movement skills because they are not certain on how to provide adequate teaching programmes (Decorby, Halas, Dixon, Wintrup, & Janzen, 2010).

2.7 Conclusions

The purpose of this chapter was to investigate FMSs and highlight the role they play in the development of motor skills in children. The challenges of teaching and learning FMSs were highlighted giving a detailed discussion of the lack of development of FMSs in children. It is important for children to develop these skills if they are to progress to more complex movements in sports and daily physical activity. It has been shown that for the proper development of FMSs, children need to be given sufficient opportunities for learning and practising movement skills.

The development of movement skills begins with a new born baby as he/she gains control of movements through reflexes. This development continues as the child gets older and begins to learn more complex movements from FMSs to specialised movements. It is important to understand the process of motor development in a child as this identifies when the child is ready to learn movement skills. Motor development is a sequential process that allows children to master the simple movements before moving to more complex movements. In order for children to learn movement skills correctly, they need to be able to understand how the movements are performed and have a visual idea of how to perform them. The process of motor learning allows the child to master movement skills through practice and continual feedback on the performance of the skills.

The focus of this chapter was on the development of FMSs which are in the fundamental phase of motor development. The different movement skills discussed identify specific sub-skills that can be observed when the child is performing the task. Teachers can give valuable feedback on the performance of the skills and point out where the child could improve. To keep children involved and engaged in the learning of FMSs, innovative games and play that help develop movement skills for children need to be implemented. The attraction of games to children gives them motivation to get involved and participate in the tasks that will improve their movement skills. Teachers often use a variety of equipment to help children learn FMSs. Children can use bean bags to learn throwing, balls to learn kicking and small bats to learn striking using hands. Teachers can also incorporate music by allowing children to skip to a beat and encourage rhythmic jumping or hopping.

There is a concern that children are not being taught how to perform the right movement skills, and unstructured play and games are being used to help children develop these skills. Children are being taught awareness of the skill rather than being taught performance of the skill to attain mastery. Although free play and games are attractive to children, children cannot learn the components of a skill without a structured plan on how to teach these skills. Teachers need to be involved in observing how the child performs a skill. The child should also be given feedback on their performance in general, but also with regards to the sub-skill parts of the skill they need to improve on.

The obstacles facing teachers in instructing and assessing movement skills is that they have limited time, insufficient equipment and many teachers are not confident enough to teach FMSs. FMSs take time to develop and children need to be given a sufficient amount of time to learn and master these skills. Some FMSs need equipment like bean bags, footballs, cricket bats and skipping ropes for children to demonstrate skill performance. The challenge for teachers is to implement lessons that the children find fun and also be able to incorporate the learning of movement skills. To address this problem, intervention programmes are being carried out to educate teachers on the importance of teaching correct movement skills. Teachers are being told to focus on the sub-skill parts of the skill and observe the child’s performance. The teachers also need to be supplied with sufficient equipment to help teach FMSs. Teachers can use the equipment to create lesson plans and interactive movement to help children learn FMSs. A possible solution to this problem would be to provide teachers with technological tools that will assist them and help children learn FMSs.

The next chapter will discuss Natural User Interfaces (NUIs), what they are and how they are used. Since NUIs focus on how users interact naturally with their environment, it is possible to detect the movement made by children. The main focus will be on the Natural User Interface supported by Kinect, the features it has and how it is used generally.

Chapter 3: Natural User Interfaces

3.1 Introduction

Humans have been using user interfaces to interact with computers in order to complete tasks and receive feedback since the early stages of computing. The evolution of computers from mainframe computers to personal computers meant that user interfaces needed to be more usable and intuitive for users. The advancement of user interfaces was also due to the growth of the relationship between users and computers from many users interacting with one computer during the mainframe era, to one computer per user in the personal computing era (Harper, Rodden, Rogers, & Sellen, 2008). The relationship has grown further to include one user interacting with many computers. This has led to human computer interaction designers and researchers to design user interfaces that can be effectively used and considered natural for the user to learn. Figure 3.1 shows the evolution of user interfaces from command line interfaces to natural user interfaces, excluding the initial batch interfaces.

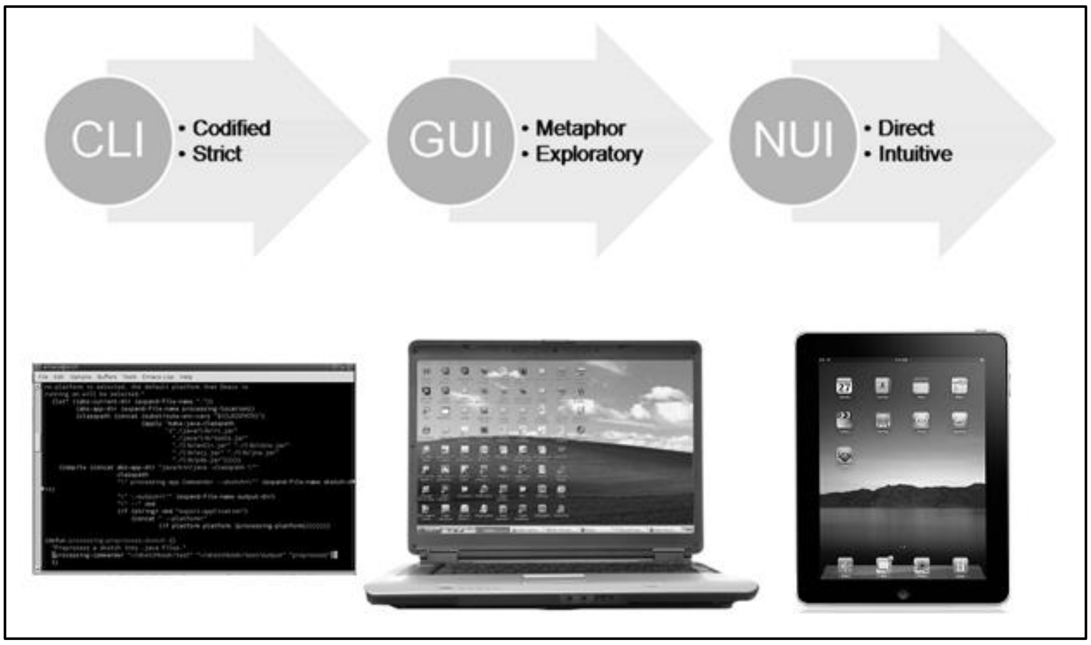


Figure 3.1: Evolution of user interfaces (Rees, 2010)

During the early stages of computing, interaction with the computer was done using a set of instructions on a punch card. This was called the batch era. The batch interface used punch cards as input devices. The punch cards consisted of a specific pattern of punched numbers that the computer could recognise. The cards were first prepared with a set of instructions describing the program and the dataset before being inserted in the computer for processing (Butow, 2007).

The next phase after the batch era was the **command line interface (CLI)**, which used the keyboard for typing commands for purposes such as word processing, accessing network resources and system administration (Blake, 2012). CLI is a type of human computer interaction (HCI) that depends only on textual input and output with the entire display screen composed only of characters and no images. The interface is navigated by typing instructions called commands at the prompts instead of using a mouse to perform a command (Linfo, 2007).

The **Graphical User interface (GUI)** followed after the CLI, with interaction mainly based on the desk top metaphor and relying on a set of user interface elements, known as WIMP (windows, icons, menus, and pointers) (Wigdor & Wixon, 2011). The WIMP elements are used to carry out commands such as opening and closing files. Since users do not need to memorise these elements, it is much easier to use and learn them, unlike command line instructions which needed to be memorised. Because of the ease of use of GUIs, the interface has become widely used by modern end-users.

As interfaces keep evolving, an emerging form of computer interaction termed **Natural User Interface (NUI)** seeks to provide a natural and better form of human computer interaction than the GUI. NUIs focus on the natural human abilities of touch, vision, speech and motion to allow for user interaction with computers.

In this chapter, an in-depth understanding of what is meant by the term ‘Natural User Interface’ (Section 3.2), as well as a discussion of the different types of NUIs, will be presented. A review of the NUI for Kinect (Section 3.3) will describe the different features of the Kinect. The discussion will include the mode of interaction with the Kinect and details of the different approaches of how movements are recognized by the Kinect. Identifying the different approaches to gesture detection and the limitations of the tracking mechanism of the Kinect will be used to determine the type of movement that can be recognized by the Kinect. An analysis of the different FMSs which can be detected by the Kinect (Section 3.4) using the selected gesture approach will determine the FMS to be incorporated as a case study for detecting an FMS.

3.2 Natural User Interfaces

NUIs are a new generation of user interfaces that promote smart and natural interactions that are engaging (Sezgin, 2012). The NUI Group (2009) defines a Natural User Interface as:

> *"an emerging computer interaction methodology which focuses on human abilities such as touch, vision, voice, motion and higher cognitive functions such as expression, perception and recall. A natural user interface or "NUI" seeks to harness the power of a much wider breadth of communication modalities which leverage skills people gain through traditional physical interaction."*

An NUI is an emerging human computer interaction that focuses on aspects of direct interactions between people and their natural surroundings, and using higher level processes such as cognition, creativity and exploration (Alfredo, Tore, & Discepolo, 2013). To understand NUIs, it is important to understand what the term "natural" means. According to Widsor and Wixon (2011), natural is the way users interact with and feel about the product. They further claim that an NUI is an interface that allows the user to feel and act like a natural, while the product meets their needs. Therefore, NUIs are not about how to use the new input technologies such as motion traction, voice, and touch, but rather, a new way of thinking about how the user interacts with the content (Blake, 2012). NUI, seek to make it easier for users to interact with the content using the natural way of interacting with objects in the real world. The users need not only use any devices like the mouse or pen stylus to manipulate objects but they would also use fingers for dragging objects or voice to perform a task. Some of the examples and applications of NUIs include multi-touch, speech, and gestural interaction.

Multi-touch

Everyday interactions with objects are usually done using the hands, picking them up, moving them around and changing their shape. Computers and computer devices have been used by people to input and manipulate information using the keyboard, mouse and joysticks. Multi-touch devices allow users to interact with information in a natural and intuitive way. Direct interaction is one of the most important concepts of HCI because it enhances the user's experience of interaction being directly with the object of interest than through an intermediate like a mouse (Yampolskiy & Govindaraju, 2007).

Multi-touch interfaces allow the user to interact with controls and applications more intuitively than a cursor-based interface, making it a more direct interaction (SkyNETitsolutions, 2011). They are input devices that are able to identify two or more simultaneous touches, making it possible for one or two users to interact with applications using gestures created by the fingers. Multi-touch devices (Figure 3.2) allow the users to swipe, pinch, rotate and perform other actions that make for better and immediate interaction with the digital content (Educause, 2008). Pinching, zooming and dragging using your fingers on the iPhone to manipulate a photo is one of the examples of natural interactions.

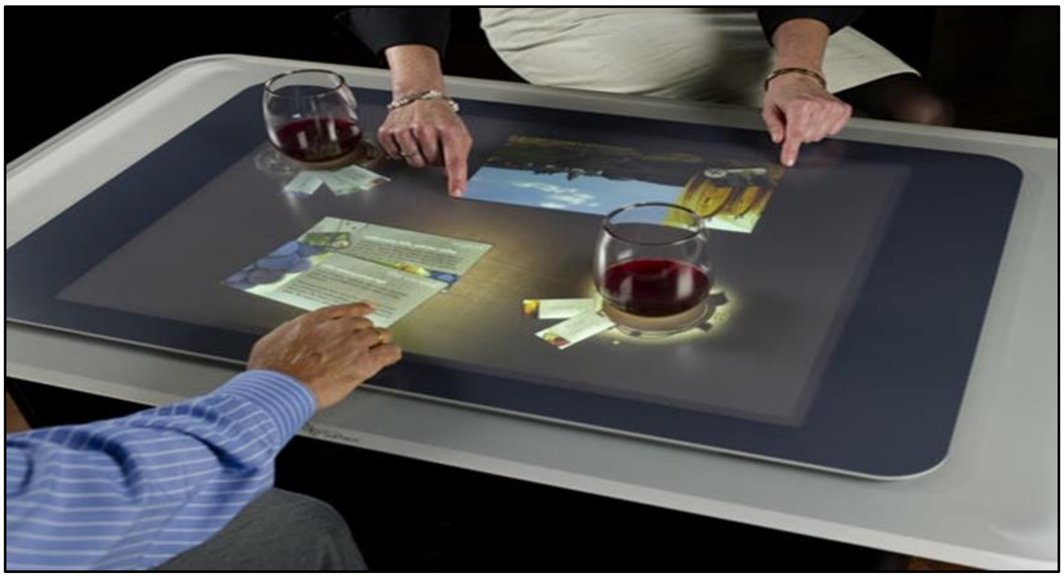


Figure 3.2: Multi-touch device

Speech Interfaces

Speech recognition systems make it possible for users to interact with the system using spoken commands. These systems are able to identify words and phrases in the spoken language and convert them to machine readable format (SkyNETitsolutions, 2011). Speech recognition is used in electronics such as tablets, mobile phones, computers and industrial electronics. Apple's siri is a speech recognition software that allows users to send messages, make phone calls and schedule meetings using spoken commands (Apple iOS, 2013). Microsoft Kinect uses speech recognition to allow users to interact with the Xbox 360 console (Microsoft Corporation, 2013a).

Gestural Interfaces

Gestural user interfaces provide new ways of interacting with devices without having any physical contact with the device. Using sensors and cameras, devices can detect direction, proximity and movements of the user interacting with the device. The interface allows the user to interact with the device using natural movements like waving (Saffer, 2014). This makes it easier for users to learn how to use the device because the interaction is mapped on to recognizable motions made in the real world. Gesture-based interfaces have become increasingly popular within the sphere of 3D interactions.

3D interactions are human-computer interactions that are characterized by spatial input in a three dimensional context, allowing users to perform tasks directly by using gestures (Bowman, Kruijff, LaViola, & Poupyrev, 2005). Console gaming systems are the most commonly identifiable 3D user interface examples that make use of 3D interactions. The **Nintendo Wii Remote** (Figure 3.3), which is a wireless handheld device, can sense 3D gestures using its 3-axis accelerometers and can sense pointing direction using an infrared camera. It contains a speaker for audio feedback, a vibration motor which provides tactile feedback and Bluetooth connectivity which is the main channel of communication (Lee, 2008). The data collected by the accelerometer and infrared camera is used as input for gesture recognition. Interaction techniques using the Wii involve a user holding the remote in one hand and directing it towards the screen. The sensor bar placed on top or below the screen contains two groups of infrared LEDs that communicate with the infrared camera on the remote. Using the Wii Remote, users can control characters and objects such as virtual golf clubs, guns or rackets in games using natural body motions (Vaughan-Nichols & Douglas, 2009)

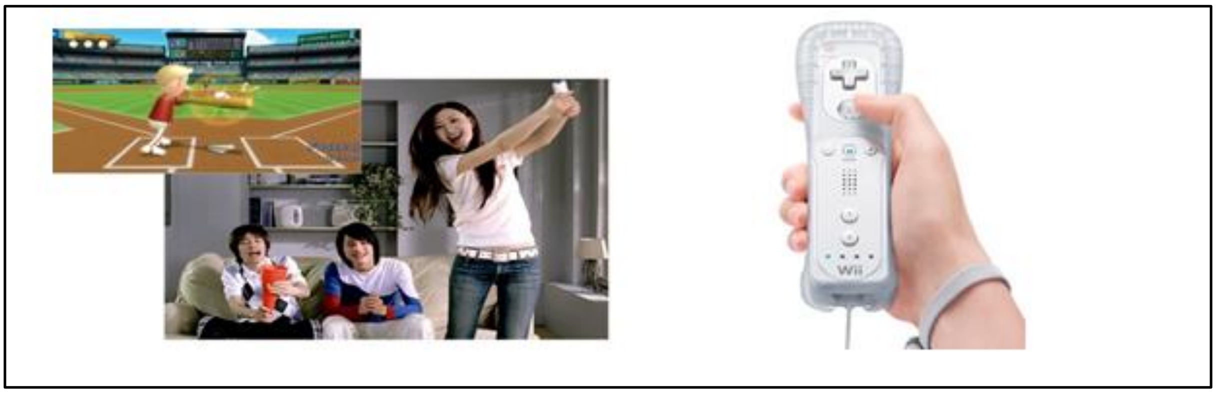


Figure 3.3: Wii Remote and 3D interaction (Otakismo, 2013; Sample.net, 2013)

Sony developed a device similar to the Wii Remote, called the **Move controller** (Figure 3.4). This device has additional sensors and capabilities for true six-degree of freedom input. The freedom of movement of a body in 3D space allows for precise forms of 3D interaction by determining the absolute 3D position and orientation of the Move controller (Bowman, McMahan, & Ragan, 2012). This means that the Move controller can determine precisely the position of the hand of the user when interacting with the interface. The Move controller contains a colour sphere attached to the wand. The colour sphere is detected by the Sony Eye camera placed on top or below the screen to determine the distance of the user. The wand uses a 3-axis accelerometer, 3-axis gyro sensor and geomagnetic sensor to determine the exact location of the wand and track motions made by the user very accurately (Tanaka et al., 2012).

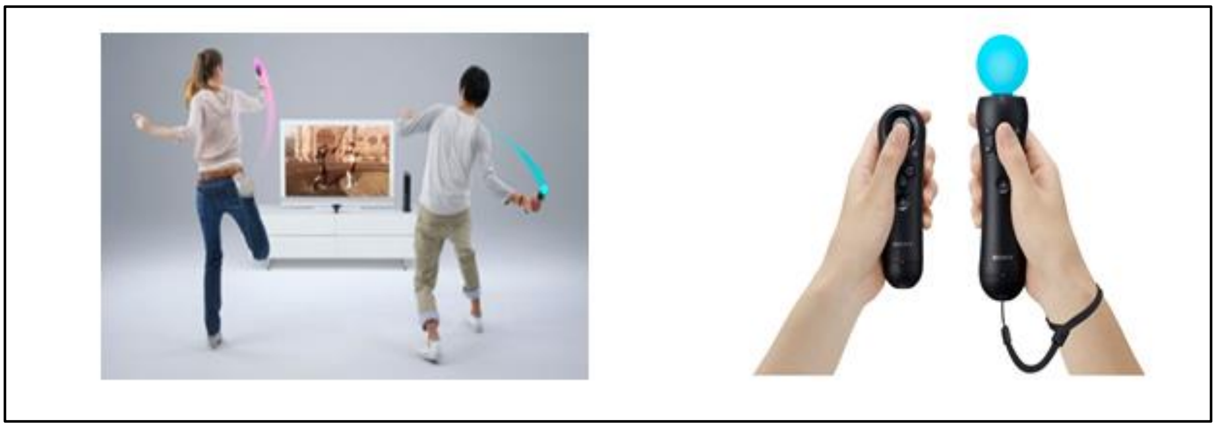


Figure 3.4: Move Controller and 3D interaction (Kalla, 2013; OLX, 2013)

Microsoft introduced the **Kinect** (Figure 3.5) which uses infrared lights and cameras to determine the 3D position and posture of the whole body without a controller, instead the user acts as the controller (Bowman et al., 2012). By using depth and RGB images from the camera, the Kinect extracts 3D information such as the position, orientation, displacement and velocity of an object. An avatar can be used to represent the user's movements with the avatar's body being controlled by gestures or movements of the user. Kinect is a full body, hands-free, motion- sensitive control system (Vaughan-Nichols & Douglas, 2009). The Kinect interface has great advantages over the Move controller and Wii remote because of its ability to be controller free. The Kinect cannot only sense users, but also objects in the environment. This allows for virtual reality applications where users can manipulate virtual objects in the real world (Tanaka et al., 2012). The Kinect can also recognize audio input using its four-microphone array to execute user commands.

Figure 3.5: Kinect and 3D interaction (Johnson, 2010; Microsoft Corporation, 2013a)

A comparison of the Nintendo Wii, Sony Move and the Microsoft Kinect (Table 3.1) shows the different capabilities and limitations of each of the devices. The Kinect has many advantages over the Wii remote and Move controller. Its ability to track full body motions and detect objects in the environments means that it can be used in a variety of situations and not just in gaming. The interface for the Kinect can be used in business meetings involving collaboration with users in different locations, such as video conferencing and also Skyping with family and friends. The Kinect has also been applied in health care to monitor physical rehabilitation of patients. The Wii remote and Move controller interfaces can track hand motions accurately but this is limited only to upper body movements. The Kinect offers a more natural way of interacting with the user than the Wii and Move. This makes the Kinect a more attractive NUI for a wide variety of users ranging from healthcare to entertainment.

Table 3.1: Comparison of the Nintendo Wii, Sony Move and Microsoft Kinect

Sensing capabilities	**Microsoft Kinect**	**Nintendo Wii**	**Sony Move**
3D information of objects in camera view	✓	✗	✗
3D information of controller	✗	✓	✓
Full body tracking	✓	✗	✗

A more detailed discussion of the NUI for Kinect (Section 3.3) will present the features of Kinect and the approaches used to implement applications to detect movements made by a user.

3.3 NUI for Kinect

Kinect is a motion sensing input device developed by Microsoft for the Xbox 360 video game console and Windows PCs. Based around a webcam-style add-on peripheral for the Xbox 360 console, it enables users to control and interact with the Xbox 360 and the PC without the need to touch a game controller, through a natural user interface using gestures and spoken commands (Boulos et al., 2011).

This makes the Kinect (Figure 3.6) a versatile device that sees people holistically using a sensor to provide ears and eyes for the device (Microsoft Corporation, 2013b). It includes several sensing hardware to detect the motion of users and the environment. The Kinect has a RGB camera that is responsible for capturing colour images and a depth sensor which uses infrared cameras to detect the depth of objects (Cervantes, Vela, & RodrÍguez, 2012). It also has a four microphone array and the components work together to provide full body tracking, as well as facial and voice recognition capabilities (Zeng & Zhang, 2012). Using depth data information of the detected object with the skeletal tracking mechanism; the Kinect is able to determine the actions of the human object. It has a motor to position the viewing angle of the camera and this can be used to track the position of a moving object.

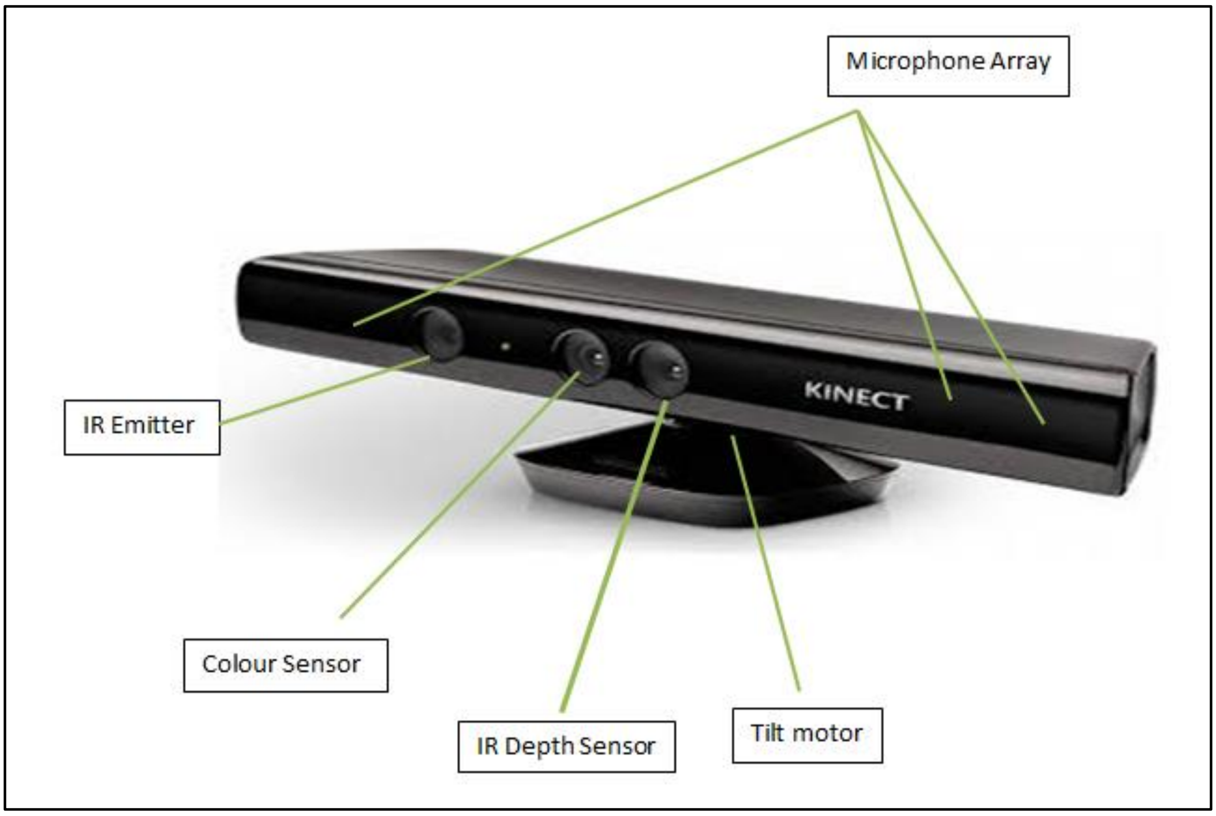


Figure 3.6: Anatomy of the Kinect (Microsoft Corporation, 2013a)

The Kinect colour sensor consists of a camera that is used to capture and stream video data. The camera detects red, blue and green colours form the source and returns a succession of still images in the form of data streams (Microsoft Corporation, 2013a). The colour stream is able to support a speed of 30 frames per second (fps) at a resolution of 640 x 480 pixels and a maximum of 1280 x 960 at 12 fps (Jana, 2012).

The primary purpose of the Kinect is the production of 3D data and this is achieved through depth image processing. This makes depth data one of the most important parts of the Kinect. The depth image processing of the Kinect not only tracks human objects but can also track non-human objects such as tables and cups (Webb & Ashley, 2012). The depth sensor of the Kinect consists of an IR emitter and an IR depth sensor which work together to determine the third dimension of an object. The IR emitter is an IR projector that emits infrared light to detect objects in front of it. When the light reflects off the object it is read by the IR depth sensor which determines the depth information of the object (Jana, 2012). This information is processed by the Kinect to produce a depth image of the detected object. The depth sensor has a field of view that determines the depth vision range. The suggested user distance from the sensor for a full body head to feet tracking is about 0.8 m to 4.0 m. For areas known as the sweet spot, where users experience optimal interactions, distances should be between 0.9 m and 3.5 m (Microsoft Corporation, 2013b). The minimum height requirement for a user is 1.0 m.

The microphone array is responsible for audio capture and detection using the Kinect. It consists of four separate microphones that are located at the bottom of the Kinect and are spread out in a linear fashion (Jana, 2012). The microphone array can be used to determine the source direction of incoming sound. It also captures quality sound by removing background noise and echoes in a room. Speech recognition is an important aspect of the Kinect. Using the microphone array, the Kinect is able to distinguish human speech very clearly by removing noise from the environment and focusing on sound in one particular direction. This allows users to provide speech commands and interact with other users online by sitting comfortably away from the Kinect using just their voice.

Skeletal tracking is the main innovation of Kinect. It builds on the foundation of depth data information but goes beyond the depth information of an object. Skeletal tracking involves the processing of depth image data to determine the positional values of several skeleton joints of a human body (Webb & Ashley, 2012). The human body is represented by a number of joints representing body parts such as the head, shoulders, neck and knees (Figure 3.7) (Zeng & Zhang, 2012). The tracking of joints provides X, Y and Z coordinate values for each of the tracked skeletal joints of the body. The X and Y coordinate points provide the position of the tracked joint while the Z value provides the distance of the joint from the Kinect. Kinect is able to track two active people simultaneously, meaning that it provides the full skeleton joints for the two users. It is able to sense up to six people but only provides an overall position for the non-active users (Microsoft Corporation, 2013b). Tracking the joints of all six skeletons would require too much processing for the Kinect to handle and this would impact on performance. In order for the Kinect to determine that the observed object is human, it makes use of machine learned data of different body shapes, clothing and poses. The Kinect is able to track 20 joints of the skeleton when the user is standing and 10 joints when seated (Microsoft Corporation, 2013b). Although the Kinect can track full body motions of users, it cannot obtain information about the orientation of the head or hands, and neither can it determine the position of the fingers of the users (Cervantes, et al., 2011). Furthermore, users should not wear clothing that causes drastic changes to their body shape as this might confuse the skeleton tracking.

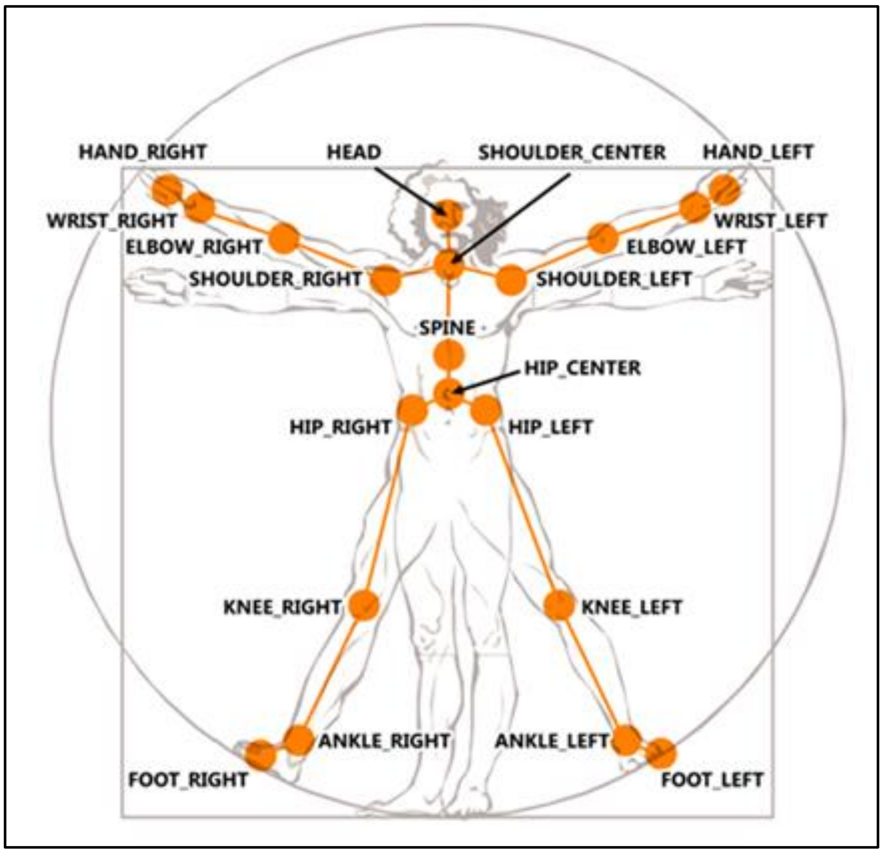


Figure 3.7: Skeletal joints tracked by the Kinect (Malcolm, 2011)

Gesture Recognition with Kinect

Gesture recognition involves the recognition of meaningful expressions of motion by a human. These could include hand, arms, head and also full body motions. These expressions can be used to communicate a message to a receiver or can be a means of interacting with the environment (Webb & Ashley, 2012). With the introduction of the Kinect, full body gestures can be detected with the use of cameras and depth sensors. The Kinect interprets gestures made by users and lets the application know what the user wants to do (Jana, 2012). An example of a gesture could be a user waving to the Kinect device. The action of waving acts as an input for the Kinect device and the application will then respond by executing certain functions. Gesture recognition forms the basis of Natural User Interface for Kinect as the users do not need to physically interact with the device using any physical connection.

The Kinect software development kit (SDK) does not provide a gesture recognition engine. Gesture recognition needs to be defined and implemented according to the requirements of the application. To recognize user gestures for the Kinect, it is left to the programmer to determine the best approach to follow when implementing gesture recognition algorithms. There are three gesture recognition approaches to detect Kinect gestures that programmers can use. These include the algorithm approach, neural network approach and exemplar or template matching approach (Webb & Ashley, 2012). The choice of the approach to be chosen will depend on the requirements of the application and complexity of the gesture to be implemented.

The **Algorithmic approach** (Figure 3.8) is used when a set of defining rules, conditions and parameters are to be met for a gesture to be valid (Webb & Ashley, 2012). Algorithms are formulated steps that are to be followed in order to solve a problem. Gesture recognition for the Kinect makes use of algorithms to detect gestures that can be broken down to non-complex movements. The direct nature approach of algorithms limits the type of gestures that can be detected using this approach (Jana, 2012). Complex movement like throwing, swinging or catching would be difficult to implement using the algorithmic approach due to the nature of their movements. The algorithmic approach to gesture recognition is recommended if the gesture to be recognized has multiple conditions which need to be validated. The gesture detection process is satisfied when the conditions are met, and the gesture is validated as either performed or not.

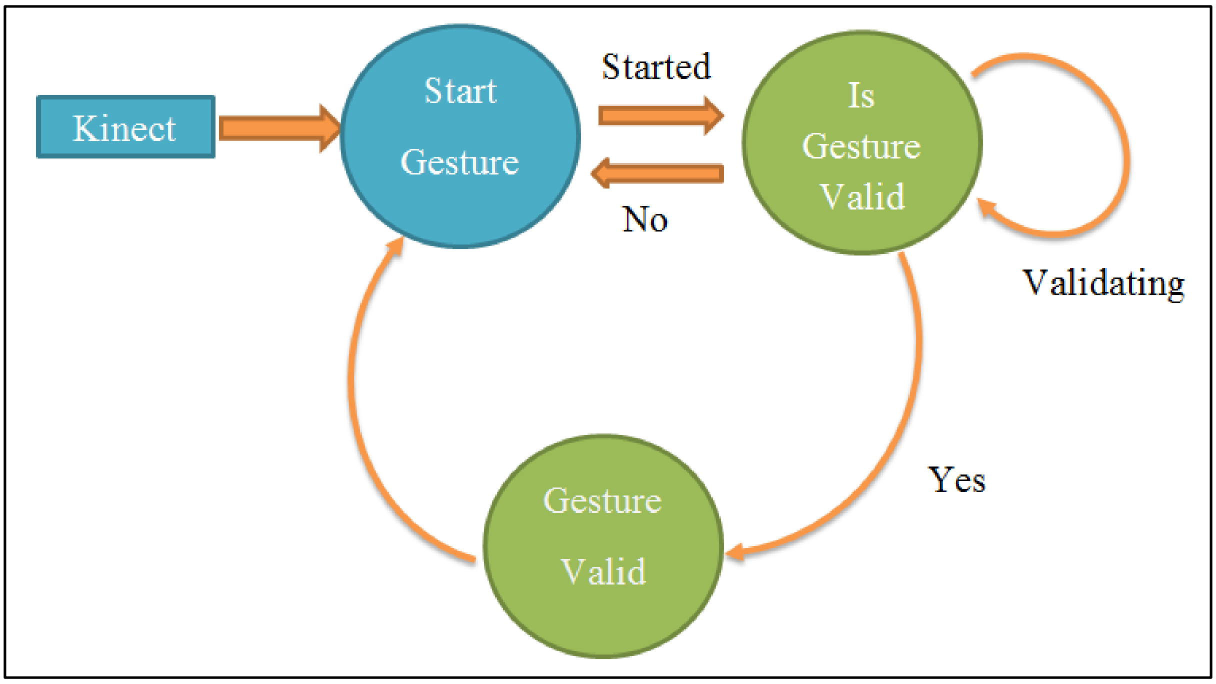


Figure 3.8: Algorithm approach to gesture detection

The **Neural network (NN) approach** is an advanced gesture detection approach used for more complex gestures. An NN is an information processing pattern that consists of processing units called nodes which are interconnected together so that they can solve specific problems. NNs are modelled after the behaviour of biological nervous systems like the brain and how they process information using neurons. The learning of NNs, like the human brain, consists of example patterns supplied to it (Stergiou & Siganos, 1996). NNs are used to solve problems relating to pattern recognition and classification. The design (Figure 3.9) of the NN consists of an input layer which takes in information, the middle layer which is the decision maker and the output layer which outputs the outcome of the process. The choice of using NNs is entirely dependent on the complexity of the gesture. Gestures which are really complex and change as the user performs them or users who perform similar gestures differently require the use of NN. Each node of the NN is an algorithm that evaluates the different elements of a gesture. An NN requires a number of inputs and parameters for each node that grows with each outcome from the given node. The NN approach uses probabilities and statistics rather than exact values to determine whether the user performed the gesture (Jana, 2012). For example, an NN would detect that the user performed eighty percent of a given gesture and did not perform twenty percent. Based on this ratio, a decision can be made as to the outcome of the gesture. The use of NNs is a naturally complex approach to gesture recognition because of the difficulty involved in configuration and modification of parameter values as any change to one node affects all the others nodes.

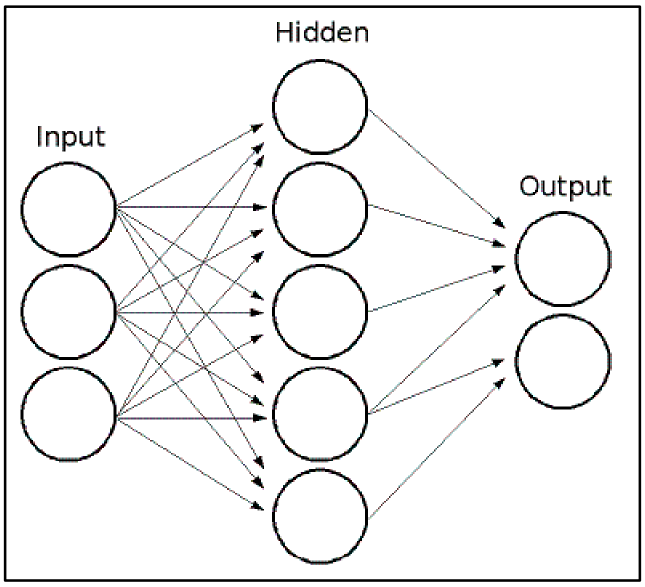


Figure 3.9: Neural network (OpenCV, 2014)

The **Exemplar or template matching approach** is based on matching the user's movements with a set of predefined gestures, and determines how close the movements are to the desired results (Webb & Ashley, 2012). Template matching is a technique that is used to classify objects by comparing a template image with its input image and finding similarities between them (Mahalakshmi, Muthaiah, Swaminathan, & Nadu, 2012). This technique makes use of machine learning by supplying the system with example data of movement to learn from. This makes the detection of gestures to be more accurate as the system compares the performed actions with the stored data. Template matching is the most suitable approach for more complex gestures and the system can easily be taught new gestures by supplying it new example templates. The desired gestures are first recorded and stored. The user actions are then validated with the stored gestures and a probability ratio between them will give the final outcome (Jana, 2012). The downside to this approach is that it is resource intensive because of the amount of data that needs to be stored and searched for during processing. Furthermore the stored data needs to have examples of different users of different body types and clothing performing the same gesture with different variations (Webb & Ashley, 2012).

3.4 Fundamental Movement Skills and Kinect

Chapter 2 discussed FMSs and the sub-skills of each FMS were identified. The sub-skills were further categorised into head, arms, knees and feet. Kinect skeletal tracking mechanism is able to track 20 joints of a full human body in the viewable area of the Kinect (Section 3.3). Recognition of gestures or movements using the Kinect requires the implementation of gesture recognition methods (Section 3.4.1). The three approaches discussed were the algorithmic approach, NN approach and template matching approach.

The choice of the approach to be used will be dependent on the nature of the gesture selected to be detected. In this section, a review of FMSs sub-skills will be presented, highlighting the sub-skills that can be recognized by the Kinect. The choice of the FMS to be incorporated as a case study is based on the FMS with the highest sub-skills that can be tracked by the Kinect. The skill to be selected will be chosen from the locomotor movement and non-loco-motor movement categories. This is because the movements in the two categories do not involve the manipulation of external objects like balls and swinging bats to be used in combination with the Kinect.

Table 3.2 shows the different categories of FMSs and the number of sub-skills that can be detected by the Kinect. The approach to be followed when implementing the selected FMS will have to determine the individual sub-skills of the FMS. Whereas the NN approach and template matching can determine that the gesture is performed, they cannot be used to determine the independent sub-skills as being performed correctly or not. Using the algorithmic approach, the algorithm determines the outcome of the gesture and its associated sub-skills. The algorithmic approach is suitable because the sub-skills can be determined and evaluated separately. The FMS to be selected should be able to be tracked by the Kinect and implemented using an algorithmic approach.

Table 3.2: Sub-skills supported by Kinect

Category	**Locomotor**						**Non-loco-motor**	
Skills	running	leaping	jumping	hopping	Gallopin	Skipping	Balance	Dodging
Hold the head still				✓		✓	✓	
legs straighten to take off			✓					
Use your arms only as needed for balance					✓		✓	
Non-support leg should be bent and not touching the support leg							✓	
Bends knee of kicking leg at least 90 degrees during the back-swing								
Knees should bend at right angles during the recovery phase	✓							
Crouch with your knees bent and			✓					
Bend knees to absorb force of landing			✓		✓	✓		
Foot on non-support leg is bent and carried behind the body				✓			✓	
Support leg should be still and foot flat on the ground.							✓	
Non-kicking foot placed beside the ball								
Kicking leg follows through high towards target area								
arms behind body			✓					
Forceful upward thrust of arms			✓					
Total	1	0	5	2	2	2	5	0

From the analysis of the Table, the FMSs that can be detected using the Kinect and implemented with the algorithmic approach are the locomotor skills of jumping, hopping and skipping on the spot. The non-locomotor skills are balance and dodging. From the locomotor skills jumping has the most sub-skills that can be detected by the Kinect. The balance skill has the most sub-skills from the non-locomotor skill that can be detected by the Kinect. Whereas the other FMSs could be detected by the Kinect with the possible exclusion of galloping and leaping, they would need to be implemented with either the template matching approach or NN. The problem identified with the FMS of galloping is that the user will end up leaving the viewable area of the Kinect when performing the skill. The skill of running would need to be modified to running on one spot for it to be detected by the Kinect and to allow the user to stay in the viewable area. The selected FMS to be implemented as a case study will be jumping. Of the skills identified, the FMS of jumping was selected because it is a locomotor skill and has the most sub-skills that can be tracked by the Kinect.

3.5 Conclusion

NUIs are changing the way users interact with computers. With the current increase in pervasive computing, the old way of interacting with computers using command line or GUI based interfaces would not be appropriate. NUIs offer a new way of HCI which allows the user to use any computing device anytime, anywhere without the need to learn how to use it. This is important because it allows any user with limited amount of experience with computing devices to interact with any computer using their natural actions as they would when performing real world tasks.

The Kinect with its full body motion capture and speech recognition provides for a natural interaction with computers. Originally thought to be a great innovation for the gaming industry only, the Kinect has proven to be revolutionary in applications ranging from healthcare to education (Zeng & Zhang, 2012). It is important for developers and designers to understand the workings of the Kinect in order to develop better and more intuitive applications. Understanding the Kinect’s colour, depth and skeletal tracking is the basis of developing applications for the Kinect (Jana, 2012). This provides the technical understanding of the Kinect and identifies the tools that developers and designers can use to develop NUIs for the Kinect. To make applications interactive to users and make them feel natural, the Kinect needs to know what the actions performed mean. Gesture recognition then makes the Kinect an NUI device. Developers need to understand the different approaches to gesture recognition to make rich and immersive NUI applications.

Gesture recognition using the NUI for Kinect can be used to detect or mimic FMSs, making the Kinect an appropriate tool to help children learn FMSs. Based on the limitations of the Kinect and the type of gesture recognition approach to be used, the jump FMS was selected as the FMS to be incorporated and tested. An architecture will be designed and used to implement the jump FMS. The design of the architecture will allow for the easy addition of future FMSs to the application.

The following chapter will discuss the design and implementation of the NUI architecture and how the jump FMS was implemented using the architecture. The chapter discusses the choice of architectural pattern to be used in the design of the architecture.

Chapter 4: NUI architecture

4.1 Introduction

Children need to develop a number of FMSs in order for them to become proficient in more advanced forms of athletics and sports. To achieve proficiency in each FMS, a specific set of observable sub-skills (sub-skills of an FMS) need to be practised (Chapter 2). Many of these sub-skills can be measured by the Kinect, an application of an NUI (Chapter 3).

This chapter aims to propose a design of an architecture using the NUI supported by Kinect to facilitate the detection of an FMS. An architecture is the foundation for any software system and provides the blue print for the implementation of a system. Software architectures (Section 4.2) and the different architectural styles (Section 4.2.1) available are investigated. The investigation will lead to the selection of a style that would be appropriate for the design of an architecture to support the detection of an FMS using the NUI supported by Kinect. The selected architectural style(s) will be used as a basic guideline when designing the architecture (Section 4.3). The architecture will be evaluated by implementing a framework (Section 4.4) that can support the implementation of different FMSs. The robustness of the framework will be tested by implementing an instance of a selected FMS to be used as a case study (Section 4.5). The designed architecture needs to demonstrate feasibility and reusability to allow for the future addition of FMSs and therefore make the architecture extensible. The chapter will conclude (Section 4.6) with a discussion of the proposed architecture.

4.2 Software Architecture

The software architecture of a computing system can be defined as the structure or structures of the system which consists of elements, externally visible components of those elements, and how they relate to each other (Bass, Clements, & Kazman, 2003). The implementation of a software system does not begin until the software architecture is complete, meeting all requirements of the system (Schmidt, 2013). It is advisable to understand the full scope of the problem and design constraints of the proposed system before implementation of the system can begin. The architecture provides the blue print of the software product, outlining the organization and behaviour of the system. This provides a guide for the development of the system and the procedure to be followed during the design process. It also forms the foundation for the evolution of the system due to changing requirements and additional functionality (Breivold, 2011).

An architecture typically consists of components that communicate with other components within the system using an interface to offer a service. Components are units of software that perform certain functions. Examples of components include objects, programmes and processes. Connectors are abstract mechanisms that facilitate communication and cooperation between components in the architecture. They ensure that information in the form of data elements is exchanged between components without altering the content of the data (Fielding, 2000). Figure 4.1 illustrates the relationship between components and connectors in a typical web architecture.

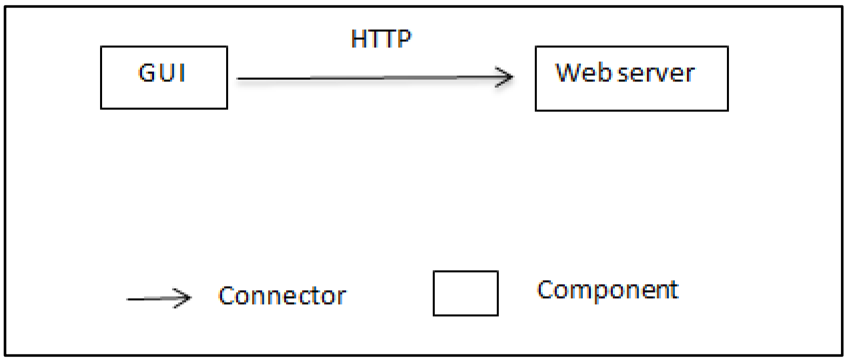


Figure 4.1: Component and Connector view

The principle of abstraction is at the heart of software architecture by hiding some details of the system using encapsulation so as to recognize and sustain system properties. The data elements consist of information that is transferred from a component, or received by a component by means of a connector. Examples of data include serialized objects, byte sequences and messages.

The design of the architecture needs to follow a set of guiding principles and standards. The set of guiding principles called architectural styles (Section 4.2.1) are used to solve a variety of repeating problems. Architectures can consist of different styles that can be used to implement a system. The common architectural styles will be stated in the following section, including the category of concerns for each of the styles. A selection will be made for the most appropriate architectural style or combination of styles to be used as a guideline for designing the NUI architecture.

4.2.1 Architectural Styles

Architectural styles are a set of principles that provide an abstract framework for a family of systems. They improve partitioning and encourage design re-use by providing solutions to repetitive problems that occur often (Microsoft Developer Network, 2014). Understanding the different architectural styles can give guidelines on the most appropriate style to implement for a system. Common styles include layered, object-oriented, client/server, component-based, domain-driven, N-tier/3-tier, and service-oriented architectures. The stated architectures can be grouped into different major focus areas that a system supports. For example, the client/server and N-tier/3-tiers architectures can be grouped in the deployment category. This category is concerned with the deployment of a system on individual or physically separate computers. The layered, component-based and object-oriented architectures can be grouped into the structure category. This category is concerned with how the application is structured, focusing on separation of concerns and division of responsibilities for a system. The domain-driven architecture is focused on defining business objects within a business domain and service-oriented architecture is concerned with communication of services by providing application functionality as a set of services (Bianco, Kotermansk, & Merson, 2007; Hoffmann, 2009).

4.2.1.1 Requirements for NUI architecture

The architecture to be designed for the detection of FMSs using the NUI supported by Kinect should be plug and play to allow for the addition of an FMS. To achieve the requirements of a scalable architecture, the design to be proposed should allow for the separation of concerns to clearly organise related functionality. The separation of concerns would allow for changes to be made without affecting the functionality of the whole application. The communication of messages should also be made clear and be loosely coupled. The proposed NUI architecture should be able to separate the collection of data and the processing of the data to determine the outcome of movement performed by the user. The layered architecture (Section 4.2.1.2) would provide for a structured framework for the NUI architecture and would be appropriate for separating the functionality of the data collection from the functionality of recognizing the movement and data storage

The proposed architecture should also allow for the system to be extendable. The architecture should be able to make the system flexible and re-usable. The object-oriented architectural style (Section 4.2.1.3) would be appropriate for managing individual FMS functionality and the relationship between the sub-skills.

The design of the NUI architecture (Section 4.3) will make use of the combination of layered and object-oriented architectures. The layered architecture will be the foundation for separating the functions related to the user interface, the detection of the movement and the data manipulation. The objected-oriented architecture will be used to facilitate the development of a system for validation of individual movement skills; for example, the jump and any sub-skills related to the jump.

4.2.1.2 Layered Architectures

Layered architectures are software structures that consist of groupings of related functionality within a system into distinct layers that communicate with one another. Layers are intended to bring quality attributes of modifiability to the system through specified communication interfaces, with a layer being an application of the principle of information hiding (Alatalo, Jarvenojal, Karvonen, Keronen, & Kuvaja, 2002). This allows for the separation of responsibility and common roles to be confined to a single layer of the system. The layered architectural style can then allow for modification of one layer without affecting the other layers. This support of the separation of responsibilities within the system makes the system flexible and maintainable (Microsoft Developer Network, 2014). A higher layer uses the services produced by the layer or layers below it. For a strict layered architecture, services are passed from a higher layer to the layer directly below it. The lower layers do not depend on the higher layers, making it easier for them to be modified and re-used in a different context. In a less strict architecture, the higher layers can communicate within the same level or with any of the lower layers. An example of a layered architecture (Figure 4.2) could consist of the presentation layer with functionality related to the user interface, the business layer with emphasis on the functioning and process of the business logic and the data layer with functionality related to the data storage and access.

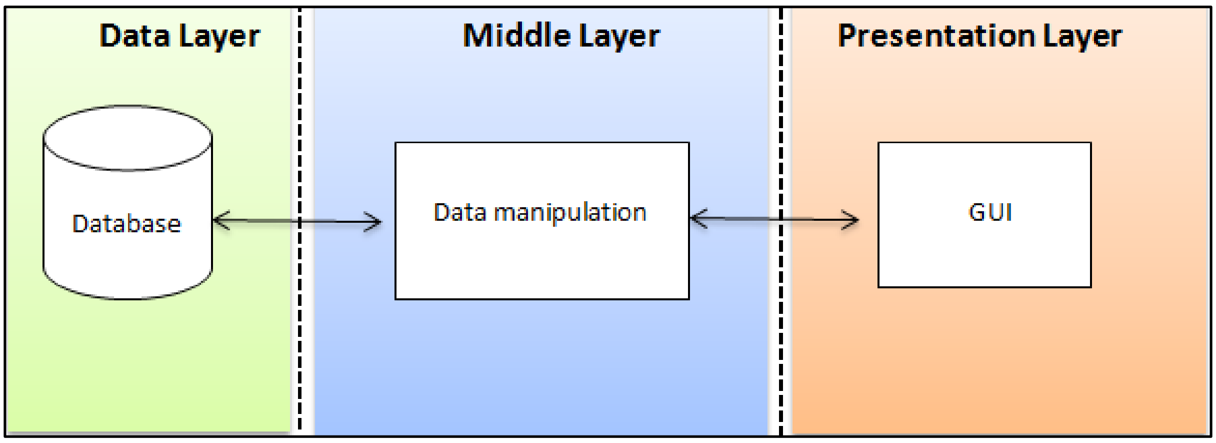


Figure 4.2: Layered Architecture

Layered architectures provide a structure that is clear and simplified by providing sufficient information to understand the role and responsibilities of the individual layers and how they interact with one another (Alatalo et al., 2002). The distinct layers provide abstraction of the view of the whole system and make it easier to understand the system by those who are not familiar with the detailed design. This concept of having independent layers allows different designers to make changes to a specific layer without knowing any information about the other layers. It also supports encapsulation as methods and properties are not exposed to the other layers. Layered architectures offer the benefit of high cohesion and low coupling because of well-defined boundaries with each layer having functionality related only to one layer. Communication between layers is based on abstraction (Microsoft Developer Network, 2014).

4.2.1.3 Object- Oriented Architectures

The main focus of object-oriented architectures is to ensure the separation of responsibilities for a system into distinct re-usable and self-sufficient objects with each object containing data and behaviour relevant to the object. An object is an entity that consists of a state and a distinct set of operations which operate on the state (Sommerville, 2001). The object's state denoting the associated attributes and operations provides services to other objects. The communication of the services is done through interfaces by accessing other objects' properties and calling methods. Objects are often modelled after real world objects and actions. Object-oriented systems can be organised into classes of objects and behaviours associated with them. An object-oriented designed system is viewed as a series of cooperating objects rather than a set of routine or procedural instructions (Microsoft Developer Network, 2014).

Objects are strongly encapsulated in that they expose functionality through properties, events and methods. They also hide the internal representation such as state and variables from other objects. By doing this, objects become easier to maintain and replace and also preserve the integrity of their internal representation (Taylor, Medvidovic, & Dashofy, 2008). Objects can also make use of polymorphism by overriding the behaviour of the base type supporting operation in the system and changing the existing code’s behaviour to a new type of behaviour. To make maintenance and updates easier, objects can inherit from other objects (Figure 4.3).

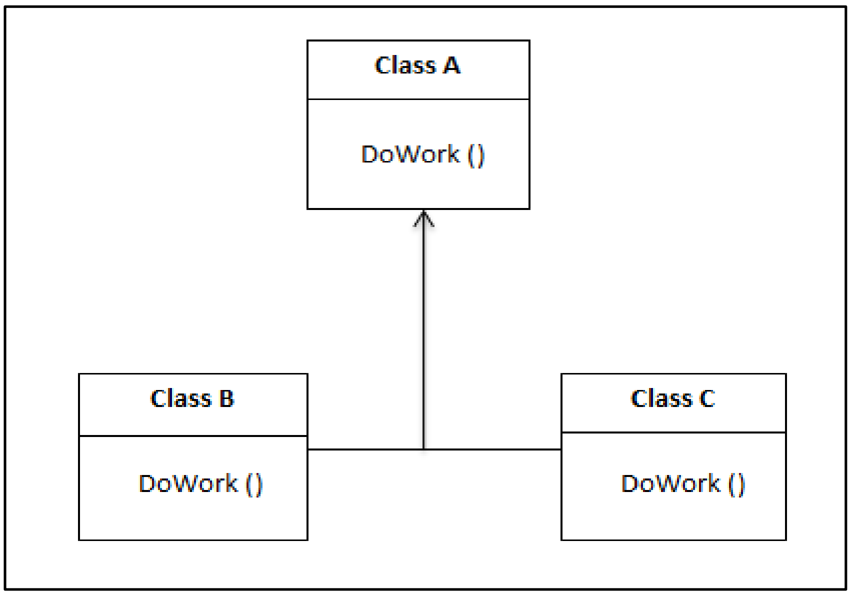


Figure 4.3: Inheritance Hierarchy

The benefits of object-oriented architectures are that by using the principles of polymorphism, encapsulation and abstraction, the system is made more re-usable and easily extensible. This allows the system to be improved by adding new functionality using the introduction of new objects or modifying existing objects (Booch, 1998). The modelling of the object on real world actions into classes and associated behaviour reduces the complexity of the system, making the structure clearer and more understandable. The integrity of the system is also preserved because data is protected and only manipulated by accessing appropriate methods (Microsoft Developer Network, 2014). It is easier to make changes to the system because changes in one class do not affect the other classes.

4.3 NUI Architecture Design

The identification of a valid FMS performance using the NUI supported by Kinect involves a three-step process. The first step involves the collection of data; the second step involves the processing of the collected data and the third step involves storing and presenting the outcome of the processed data. The design of the generic NUI architecture (Figure 4.4) can be separated into three distinct layers corresponding to the generic layered architecture design (Figure 4.2). The layers supported by the proposed architecture are the presentation layer, middle layer and data layer.

The presentation layer (Section 4.3.1) of the proposed architecture maps onto the presentation layer of the generic layered architecture. The FMS presentation layer will be responsible for the collection of data and displaying the outcome of the detected FMS on the user interface screen. The data from the presentation layer is communicated to the middle layer. The middle layer (Section 4.3.2) will implement the functionality related to the recognition of the FMS that is to be performed by the user. It corresponds to the middle layer of the generic layered architecture. The middle layer also manages functionality of sub-skills associated with the FMS being recognized. The processed data is communicated to the presentation layer and the data layer (Section 4.3.3). The data layer represents the storage of data in the generic layered architecture. The proposed data layer will store a log file of the outcome of the performed movement and the knowledge base of the relationship between an FMS and its associated sub-skills.

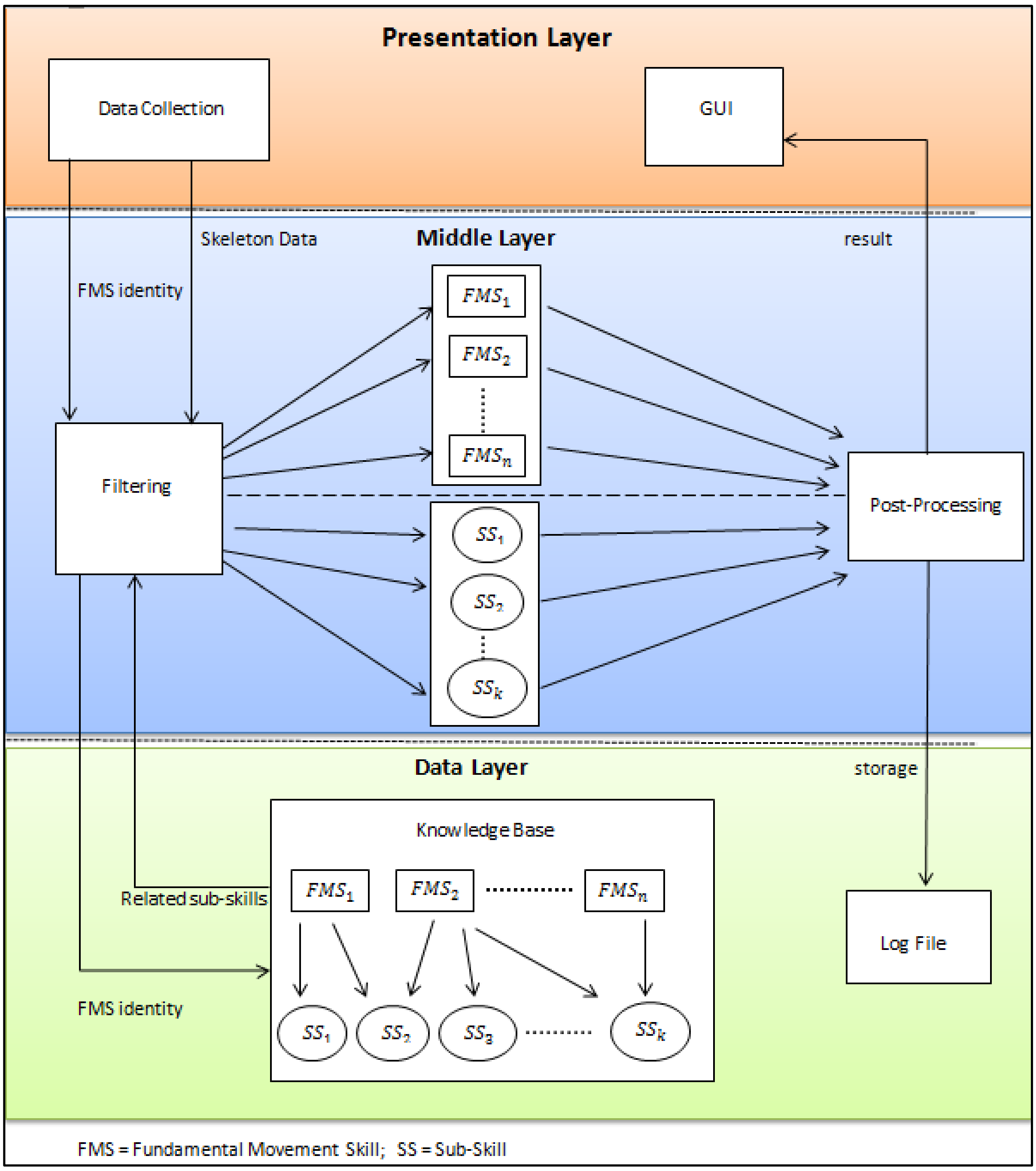


Figure 4.4: Generic architecture of FMS

The use of objects corresponds to the object-oriented architectural style. The objects of the NUI architecture are represented by FMS components FMS_1, FMS_2,…FMS_n, where *n* is the number of distinct FMSs supported by the architecture. The related sub-skills are represented by SS_1,SS_2,SS_3…SS_k, where *k* is the number of distinct sub-skills supported by the architecture. The object-oriented architectural style will allow different FMSs to access functionality related to the sub-skills they have in common with other FMSs. The architecture makes use of the object-oriented principles of encapsulation, inheritance and polymorphism to allow the sub-skills to be accessed only by components in the middle layer. The combination of the two architectural styles makes the proposed architecture modular and easily adaptable. This supports the adding of an additional FMS with its sub-skills or modifying an existing FMS. A new object can inherit available functionality or implement new functionality without affecting the functionality of the other objects. The architecture also allows for the re-use of functionality as added FMS objects can access common sub-skills that are already supported.

4.3.1 Presentation Layer

The presentation layer is the main interaction area of the system. It consists of the data collection and graphical user interface component (GUI). The data collection component is simply the collection of data of the tracked user from the Kinect and the identity of the selected FMS to be performed. The graphical user interface component presents the collected data on the screen including the outcome of the FMS performed.

The processing of recognizing a gesture begins by selecting the type of FMS to be performed by the user. Tracking the user is, however, the main task. When the Kinect detects a user, the collected data is displayed on the user interface screen as a depth image of the user. The presentation layer collects data in the form of depth images and skeletal information of the user. The depth and skeletal data are used to collect the joint information of the user that is being tracked by the Kinect. The joint information contains coordinates (x, y, z) of each joint's position value corresponding to the position of the joint movement of the user. The collected data from the Kinect is sent to the middle layer (Section 4.3.2) in the form of skeleton data of the user which contains a collection of all the tracked data joints of the user. The identity of the FMS to be performed by the user is also retrieved from the data collection component and sent to the middle layer.

When the gesture performed by the user is recognized by the middle layer, the outcome of the processed data is sent to the presentation layer for display. The data received consists of the outcome of the performed FMS and its associated sub-skills. The details displayed on the user interface screen include the results of the performed movement as having been a successful or failed attempt. The details will also indicate whether the sub-skills were performed correctly or not. This provides immediate feedback to users allowing them to view sub-skills that were not performed correctly. Figure 4.5 shows the GUI design used for the proposed NUI architecture. The layout allows the user to visualize the performance of the movement in the image view, the FMS selected to be performed in the FMS view and its associated sub-skills in the sub-skills view.

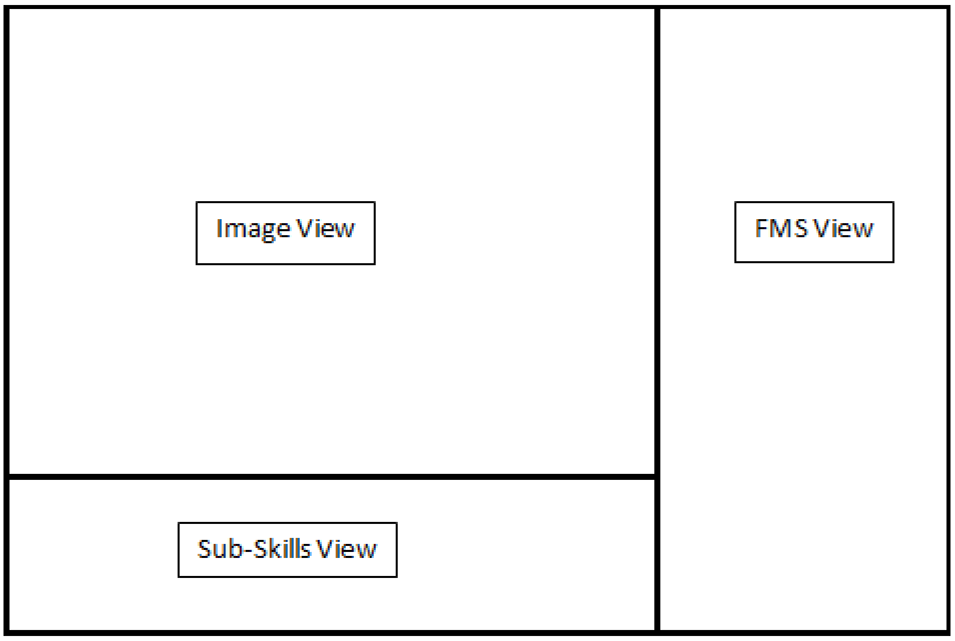


Figure 4.5: GUI Layout

4.3.2 Middle Layer

The middle layer's main functionality is the recognition of an FMS and validating the successful attempt of its associated sub-skills. It makes use of gesture detection algorithms to recognize the FMS performed by the user (Section 3.2.1). It consists of four components, namely, the filtering, FMS, sub-skill, and post-processing components. The filtering component is responsible for filtering joint information related to a particular FMS, its associated sub-skills and the identity of the selected FMS. The FMS component is responsible for checking the validity of an FMS gesture. The sub-skill component is responsible for the validation of individual sub-skills of the selected FMS. The post-processing component determines the outcome related to the performed FMS and its associated sub-skills.

The purpose of the filtering component is to filter the skeleton data information received from the data collection component in the presentation layer. It also receives the selected FMS identity and uses the identity to retrieve sub-skills (from the data layer) associated with the identified FMS. The skeleton data received from the presentation layer consists of 20 joints tracked by the Kinect. The filtering component is used to select relevant joints to be used by the FMS component to determine the outcome of an FMS. The relevant joints for specific sub-skills of the identified FMS are also determined in the filtering component. The tracked joints relevant to specific sub-skills are used to determine the outcome of the attempted sub-skills.

The FMS component is responsible for defining the algorithms to be used to detect the gesture performed by the user. The FMS component determines the requirements for implementing a detection algorithm for a specific FMS. The developer can add an FMS by implementing the algorithms corresponding to the requirements of the FMS to be detected and plugged in. The FMS component receives the relevant joint data from the filtering component. The algorithms make use of the received skeleton's coordinates (x, y, z) data joints to determine the outcome of the performed FMS. The algorithm outcome is passed on to the post-processing component for post-processing.

The purpose of the sub-skill component is to determine the outcome of the sub-skills of the identified FMS to be performed. The implemented sub-skill algorithms are used to evaluate the movement of the joints received from the filtering component when the user is performing an FMS. Each FMS has specific sub-skills that need to be performed correctly for the movement to be valid (Section 2.4.1). The outcome of the individual sub-skills component is sent to the post-processing component.

The purpose of post-processing is to determine an outcome of the performed FMS. When the selected FMS is determined, post-processing receives the computed result from the FMS component and its associated sub-skills from the sub-skill component. The post-processing component then determines whether the performed FMS is a successful attempt or not. The outcomes of the FMS and sub-skills values are then sent to the presentation layer for display. The outcome is stored by sending the information to the data layer (Section 4.3.3) for storage.

4.3.3 Data Layer

The purpose of the data layer is to store the relationship between an FMS and its associated sub-skills. Each FMS points to the sub-skills that need to be validated in order for the FMS to be performed correctly. By making use of object-oriented design, the sub-skills inherit behaviour of the main sub-skill object which determines if the skill is valid or not (Figure 4.6). The knowledge base manages all the different sub-skills. The data layer receives the

FMS identity from the filtering component in the middle layer and sends it to the knowledge base. The knowledge base then retrieves the sub-skills relevant to the identified FMS and sends the data back to the middle layer for processing.

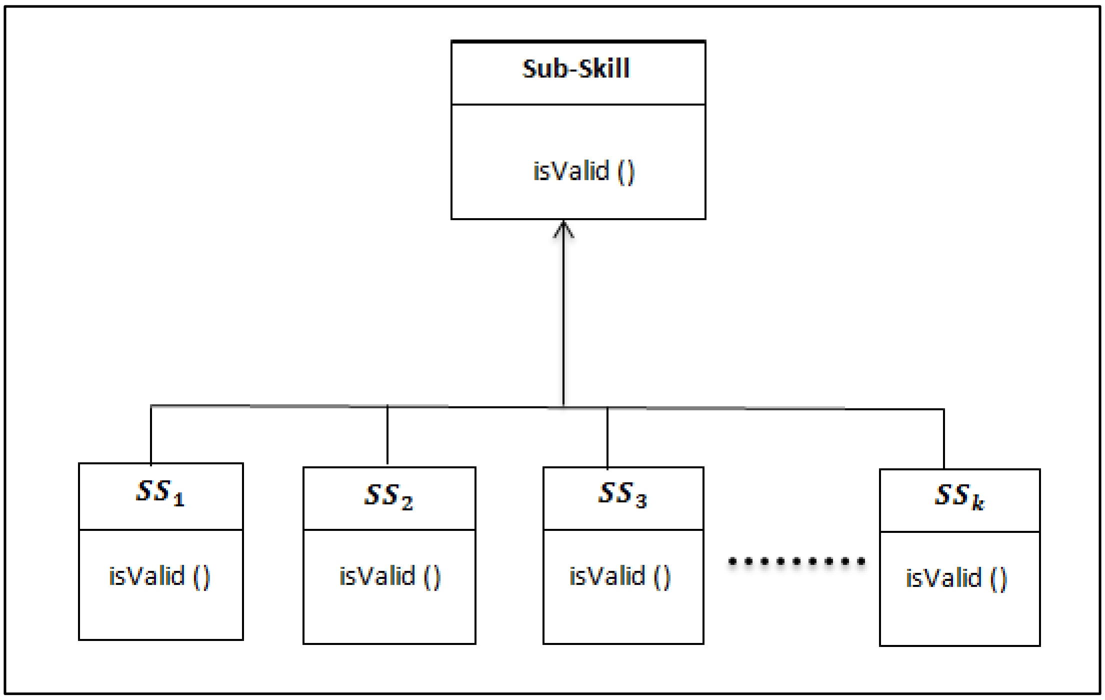


Figure 4.6: Inheritance Hierarchy

The data layer also stores the outcome of the performed FMS and its associated sub-skills. When the gesture is performed and a result is returned, the data layer receives the result through the post-processing component and stores the data in a log file. The information to be stored includes the name of the FMS that was performed and the outcome of the gesture detection result. The FMS performed will store the associated sub-skills of the identified FMS in the form of logical true or false values.

4.4 NUI Framework Implementation

This section discusses the evaluation of the proposed architecture by implementing a framework (Figure 4.7) for detecting an FMS. The first layer represents the presentation layer which is implemented using the Windows Presentation Foundation (WPF) control library. The second layer is represented by the middle layer which is implemented as a gesture detection library that manages and detects gestures. The data layer represents the third layer of the NUI architecture which is implemented as log file writer and the knowledge base.

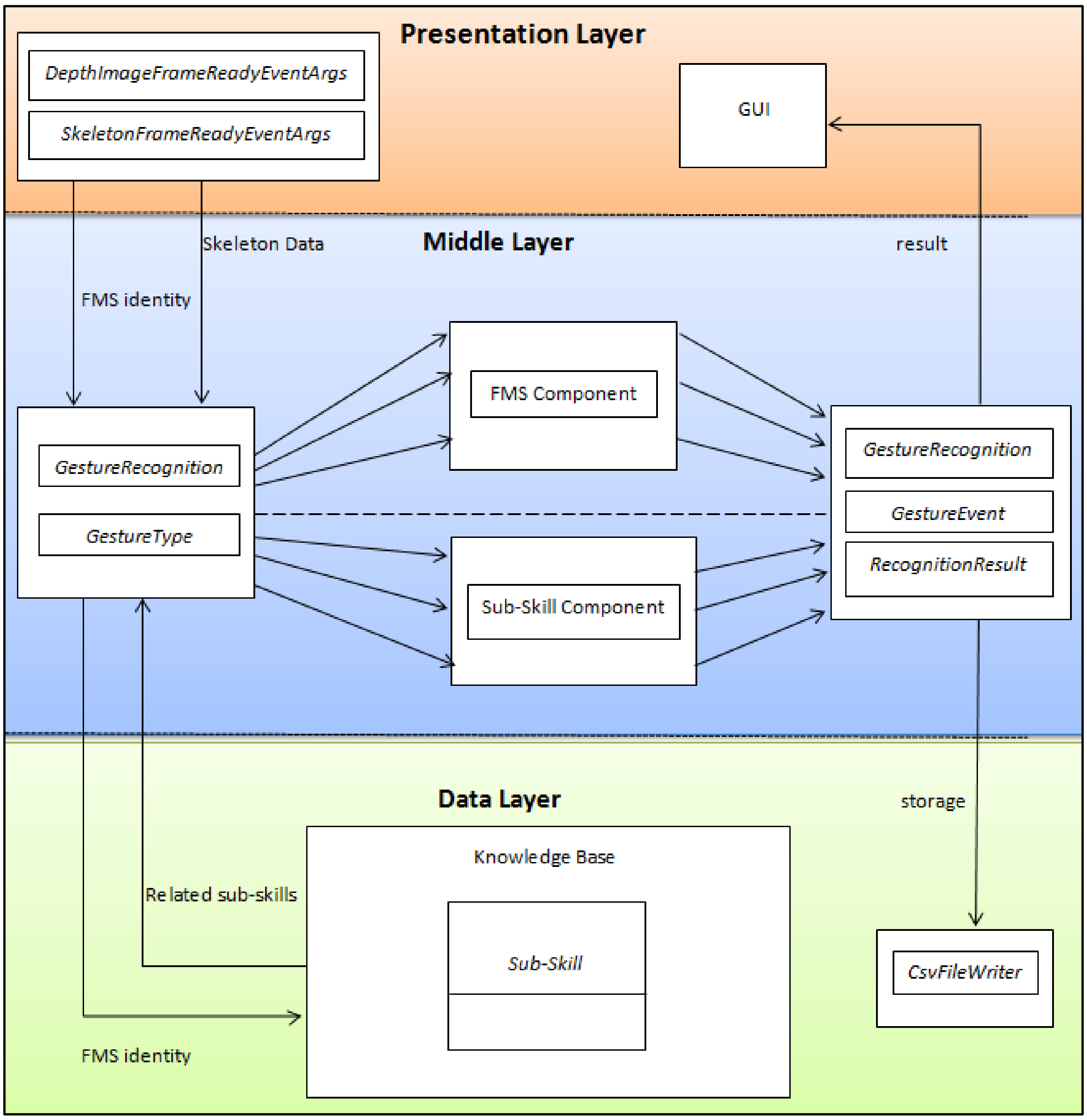


Figure 4.7: Proposed framework

4.4.1 Presentation Layer

The presentation layer is implemented on top of the middle layer and uses the WPF image control to display the data from the Kinect sensor. It implements the logic for the access of depth and skeletal data from the Kinect. The depth and skeletal data are retrieved by subscribing to either the event model or the polling model provided by the Kinect Sensor. The event model provides image frames to the application whenever a new frame becomes available from the Kinect sensor. The polling model provides image frames only when requested for by the application (Webb & Ashley, 2012). The depth frame images are obtained by enabling the depth stream and attaching an event handler *DepthFrameReady* that processes depth images whenever a new one becomes available. The depth images are retrieved from the exposed OpenDepthImageFrame of the *DepthImageFrameReadyEventArgs* class. The processed data is then integrated with the WPF image control and then rendered on the user interface screen.

To track the human body and capture skeleton joint data, the application needs to enable skeletal tracking. The skeletal data is obtained from the skeleton stream channel. Skeletal tracking retrieves skeleton frames and whenever a new skeleton frame is ready, the attached event handler *skeletonFrameReady* processes the frame data. The event handler takes as input an event argument of type *SkeletonFrameReadyEventArgs* which retrieves the current skeleton frame from the Kinect sensor. The retrieved skeleton frame object is copied into a skeleton array that can store up to six skeleton objects (Jana, 2012). The tracked skeleton that is accessed can be retrieved from the array and the information can be passed on to the middle layer. The skeleton data containing tracked joint coordinates passed on to the middle layer is used to determine the outcome of the selected FMS.

4.4.2 Middle Layer

The middle layer implements the functionality for NUI detection. It consists of various classes that can be used to determine the outcome of a performed FMS. The main class *GestureRecognition* is used to receive data from the presentation layer and communicate with the other classes. The other classes include *GestureType, GestureEvent* and *RecognitionResult* classes in the middle layer. The *GestureType* class is used to store the identifier used to relate to a particular FMS. If an FMS is added, its specific identifier is added to the *GestureType* class. The *GestureEvent* class stores the result of the gesture outcome. The result is obtained from the *RecognitionResult* class which holds the expected results.

The programmer can implement FMS classes that have functionality to determine the movement of a particular FMS gesture to communicate with the *GestureRecognition* class. The different FMS classes implement their own algorithms to determine how the FMS to be detected will be recognized. Figure 4.8 shows the skeleton with the different joints used to determine the outcome of an FMS. The implemented algorithms use the relevant skeleton joint data to calculate the outcome and determine if the movement meets the required conditions for an FMS to be detected.

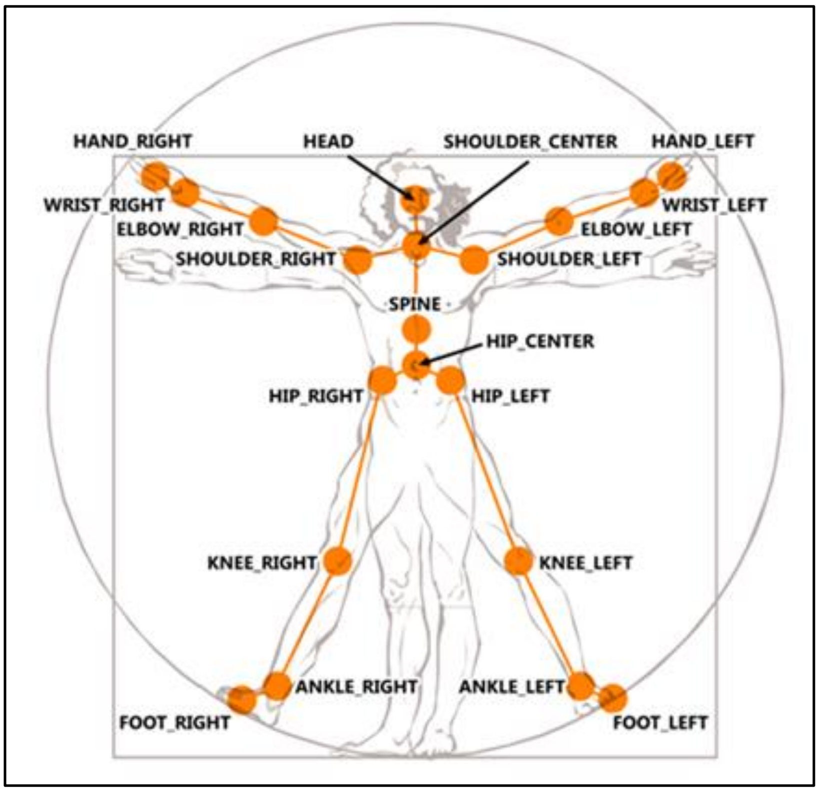


Figure 4.8: Skeleton joints used to determine FMS gesture outcome (Malcolm, 2011)

Each skeleton joint is represented as (*x*,*y*,*z*) coordinates in a three-dimensional plane. The x and y represent the joint coordinate in the plane, while the z coordinate indicates the distance of the joint from the Kinect. The distance between two different joints (x_1, y_1, z_1) and (x_2, y_2, z_z) can be calculated using the following formula:

$$\text{Distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \qquad \text{Eq.4.1}$$

The *GestureRecognition* class is responsible for the detection and validation of the FMS gestures performed by the user. It corresponds to the filtering and post-processing component in the proposed architecture. It consists of instantiated FMS gestures and provides an interface specifying which FMSs are to be detected. The FMS gesture to be detected is identified by comparing the FMS identity with the FMS type in the *GestureType* class. The *GestureRecognition* class communicates with the implemented FMS class to initiate the process of detecting the identified FMS using its defined algorithms. The FMS class calculates the outcome and uses the *GestureEvent* class to store the result. Sub-skills associated with the particular FMS are retrieved from the knowledge base in the data layer using the FMS identifier.

4.4.3 Data Layer

The data layer consists of a knowledge base class that contains the relationship between an FMS and its associated sub-skills. The *Sub_Skill* class is the base class for the validation of the specific sub-skills of a particular FMS. The class defines a virtual method to return a true value if the sub-skill was detected or a false value if the sub-skill was not detected. Defining the *Sub_Skill* as a base class allows the programmer to add more sub-skills that can inherit from the base class but implement their own behaviour by overriding the base class methods.

The data layer also stores the outcome of a particular FMS that was performed, including its associated sub-skills. It implements a *CsvFileWriter* class that writes the outcome of the performed movement to a comma separated values (CSV) file. The structure of the file consists of the FMS name that was performed, including the names of the sub-skills associated with the performed FMS (<*FMS*>, $< SS_1 >, < SS_2 >, \dots, < SS_k >$, where k is the number of distinct sub-skills). The sub-skills values show a yes value to denote a correct attempt and no value for an incorrect attempt in the CSV file.

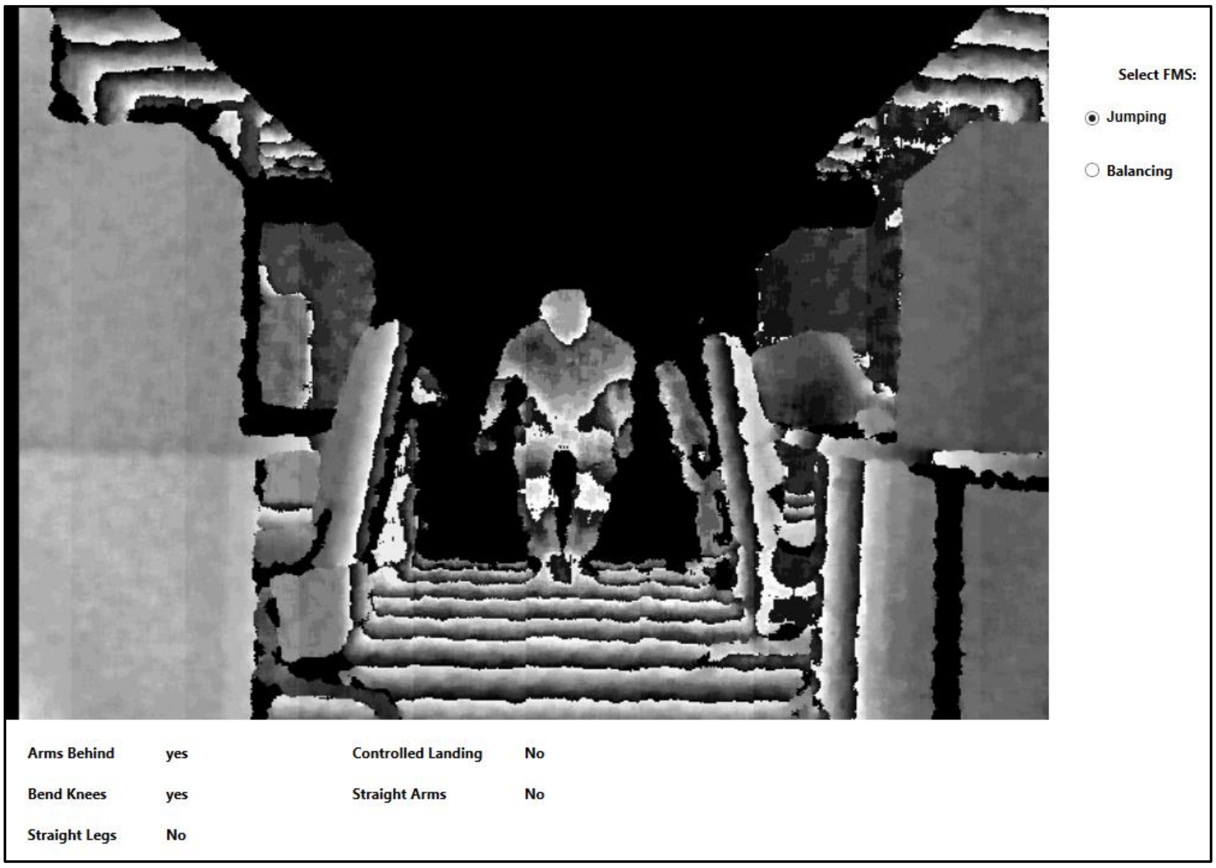
4.5 Jump Detection instance

This section discusses the implementation of an instance of an FMS to demonstrate the suitability and robustness of the framework in its ability to be used to implement an FMS. The selected FMS to be implemented is the jump because it is the most appropriate locomotor skill to be detected by the Kinect (Chapter 3).

4.5.1 Presentation Layer

The implementation of the jump makes use of the depth and skeletal data to determine the outcome of the jump. This corresponds to the data collection component of the framework. The identity of the jump identifier can be selected from a list of options in the data collection component. The skeletal data and jump identifier are retrieved from the data collection component.

The implementation of the Jump FMS uses the event model to obtain a continuous supply of data in order to determine the jump FMS and its associated sub-skills. The application will subscribe to depth and skeletal data image frames by enabling the depth and skeletal channel streams of the Kinect. The detected skeleton and the selected FMS jump identifier are sent to the middle layer to initiate the detection process (Section 4.5.2). The outcome from the middle layer is returned to the presentation layer to be displayed on the user interface screen (Figure 4.9).


Select FMS:

[✓ Jumping]

Balancing

Arms Behind	yes	Controlled Landing	No
Bend Knees	yes	Straight Arms	No
Straight Legs	No		

Figure 4.9: GUI screen layout for jump FMS

4.5.2 Middle layer

The middle layer implements functionality related to the jump detection. The filtering component filters sub-skills related to the jump including the relevant joints. The FMS component determines the outcome of an attempt at a jump. The sub-skill component determines the outcome of the associated sub-skills. The post-processing component determines the correct or incorrect outcome of a jump and its associated sub-skills.

The filtering and post-processing component is implemented by the *GestureRecognition* class. If the *GestureType* class matches the received jump identifier with the expected identifier, the process of detecting a jump is started. The *GestureRecognition* class sends the tracked skeleton data to the *JumpFMS* class. The detection function in the *JumpFMS* class returns the result of the algorithm to the *GestureRecognition* class. The FMS gesture is recognized by raising a *GestureRecognized* event of type *GestureEvent* class that passes in the *RecognitionResult* class consisting of the result of the detected jump gesture.

The *JumpFMS* class implements the functionality for the jump and the algorithms approach (Section 3.3) to detect the movement of a Jump. It defines the logic of detecting the motion of jumping. The process of jump detection begins by passing the currently tracked skeleton joints to the jump FMS class as input. The algorithm determines the jump by receiving joint X, Y and Z coordinates of the tracked skeleton joints from the *GestureRecognition* class.

To detect a jump, the *JumpFMS class* consists of the algorithm to determine the height of the user (Figure 4.10). The calculated height of the user is used to determine the action of jumping by setting a threshold value above the height of the user. To determine the height of the user, the height is calculated by summing selected joint segments. The joint segments are:

- Head to Shoulder Centre
- Shoulder centre to Spine
- Spine to Hip Centre
- Hip Centre to Knee Left or Knee Right
- Knee Left or Knee Right to Ankle Left or Ankle Right
- Ankle Left or Ankle Right to Foot Left or Foot Right.

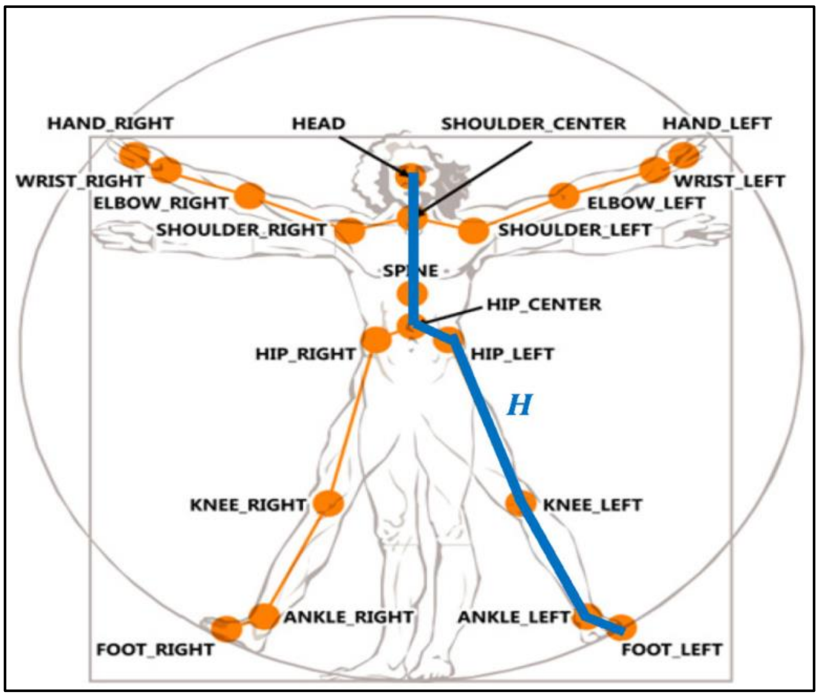


Figure 4.10: Estimated height of tracked User

The tracked skeleton from the Kinect provides coordinate joint positions (*x*, *y*, *z*) of the head joint. The height-threshold (Figure 4.11) is set by using the calculated height and adding a threshold value to the height. The gesture result will be recognized as a success if the user's head Y joint position is greater than the height-threshold Y position. When the gesture is detected as a jump attempt, the next process is to determine and validate the sub-skills associated with a jump.

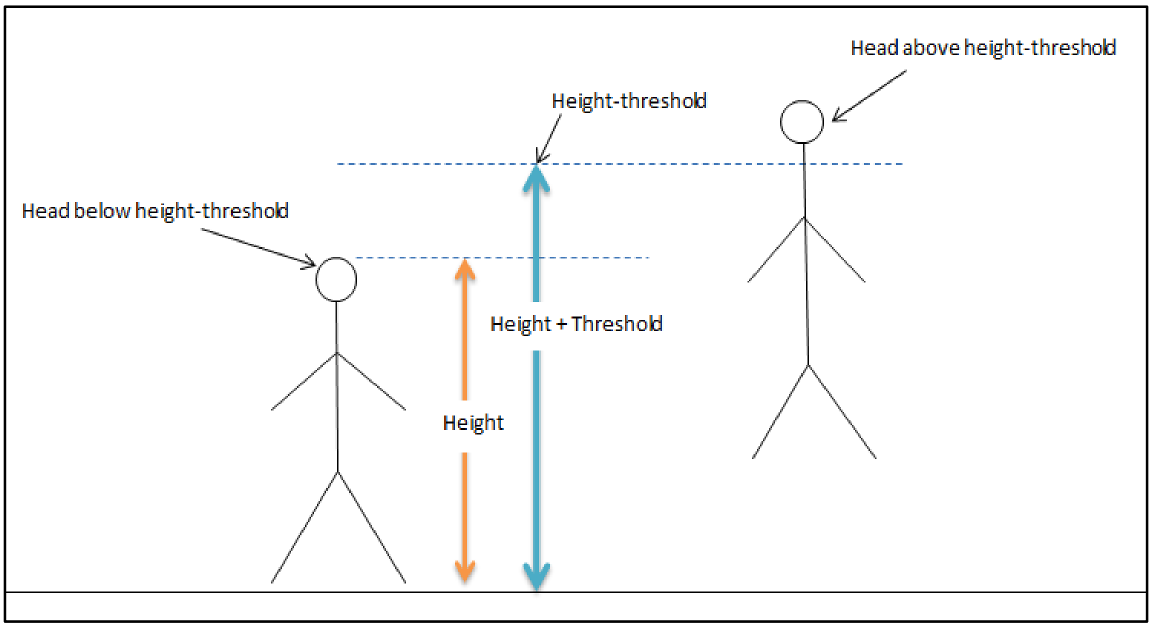


Figure 4.11: Setting height-threshold for a jump

GestureRecognition class also communicates with the knowledge base to retrieve the sub-skills associated with the jump. As the user performs the jump gesture, filtered joint information is passed on to the retrieved sub-skill classes of the jump. The sub-skill classes validate whether the user performs each of the sub-skills of the jump accurately. The *GestureRecognition* class then sends the FMS result and sub-skills validation to the presentation layer for display and writes the data to a CSV file using the *CsvFileWriter* class.

Each individual sub-skill will be evaluated by detecting whether it is performed correctly or not (returning a true or false value respectively). For example, to calculate the sub-skill *ArmsStraight*, the position of the head joint, left hand joint and right hand joint will be determined. The algorithm has to determine that the user has his/her hands stretched as high as possible. The two hand Y joint positions will have to be above the head Y joint position (Figure 4.12). The calculated length of the stretched hand will have to be more or equal to the actual hand length calculated. The length of a hand will be calculated by measuring the distance from the shoulder joint to the elbow joint and the hand joint. The application checks that all the conditions are satisfied and returns a result of the performed attempt.

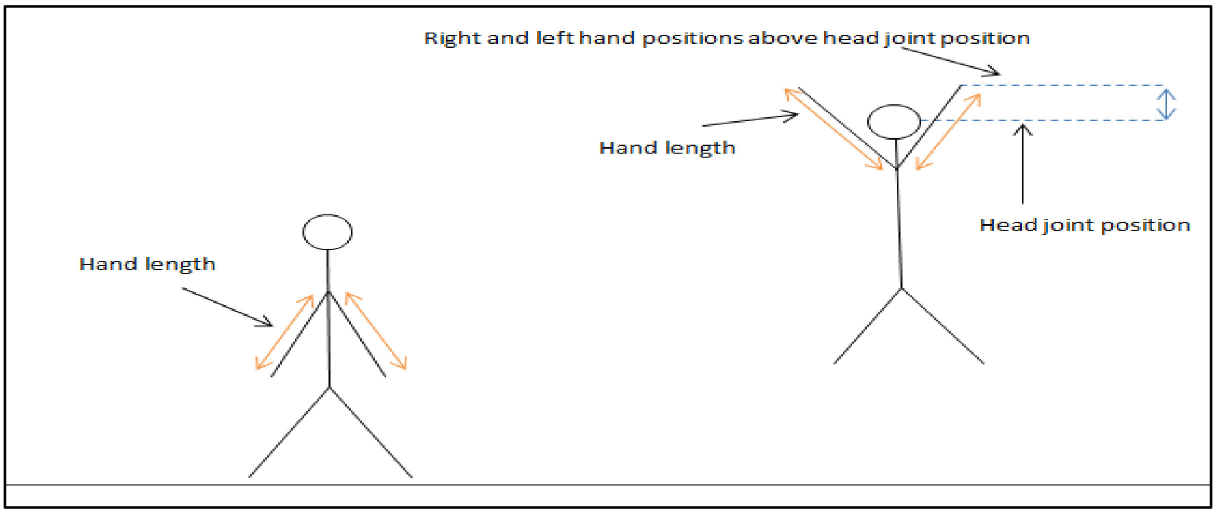


Figure 4.12: Calculation of arms straight

4.5.3 Data Layer

The knowledge base consists of implemented sub-skill classes. The jump sub-skill classes inherit functionality from base *Sub_Skill* class but implement their own behaviour using different algorithms to determine the outcome of each of the different sub-skills (Figure 4.13). There are five sub-skills associated with the jump. The classes representing the specific sub-skills include *ArmsMoveBack*, *ArmsStraight*, *LegsStraight*, and *BendKnees*, *ControlledLanding*. Each class implements its own algorithms to determine the outcome of the sub-skill validation.

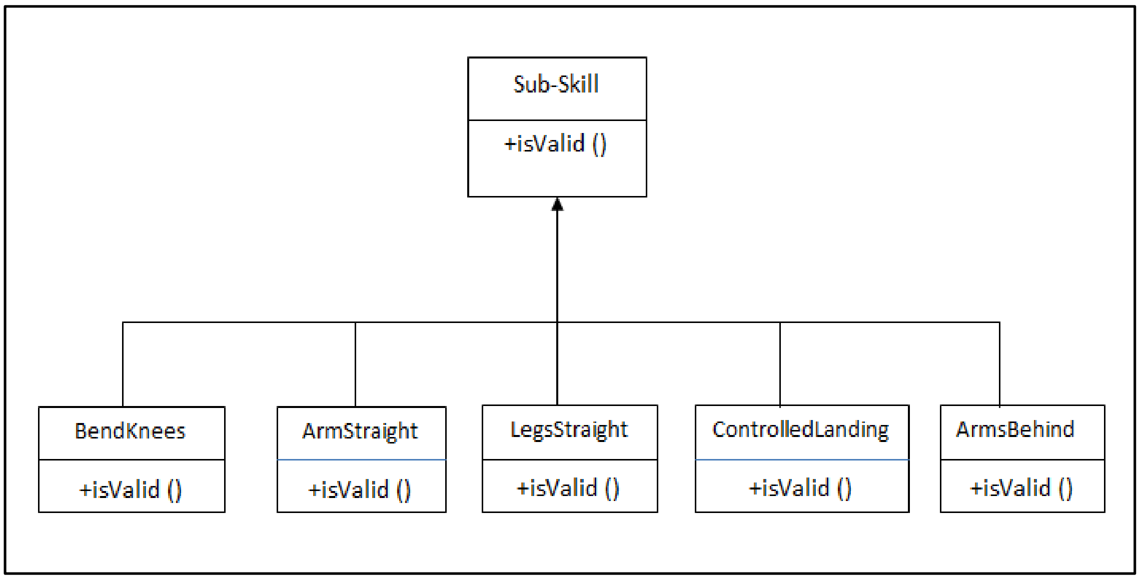
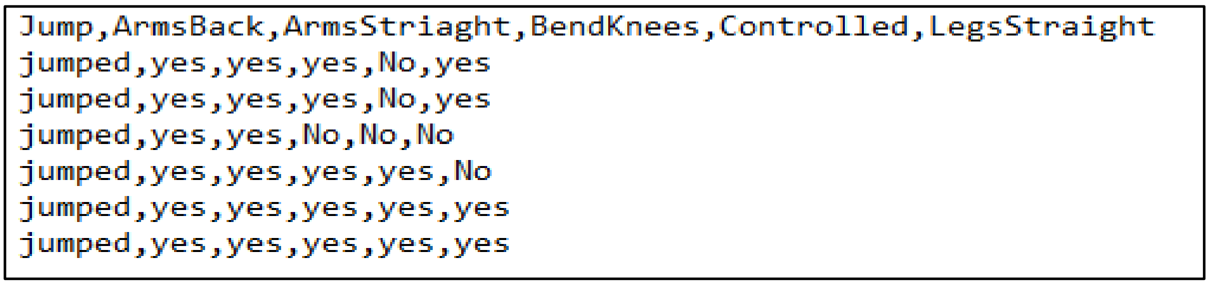


Figure 4.13: Jump Inheritance Hierarchy

The data layer stores the outcome of the jump performance in a CSV file. The structure of the file consists of the outcome of the jump and the values of the outcome of the sub-skills associated with the jump (<*Jump*>, <*ArmsMoveBack*>, <*ArmsStraight*>, <*BendKnees*>, <*ControlledLanding*>, <*LegsStraight*>).The stored results can be viewed in an Excel spreadsheet. Figure 4.14 show the details of the performed outcome of a jump attempt and its associated sub-skills.



```
Jump,ArmsBack,ArmsStriaght,BendKnees,Controlled,LegsStraight
jumped,yes,yes,yes,No,yes
jumped,yes,yes,yes,No,yes
jumped,yes,yes,No,No,No
jumped,yes,yes,yes,yes,No
jumped,yes,yes,yes,yes,yes
jumped,yes,yes,yes,yes,yes
```

Figure 4.14: Image view of results

4.6 Conclusion

To design an efficient and stable system, developers need to plan and design how the system will be developed, including the functionality. The design of the system will need to follow a structured framework to clearly communicate the function of the system. Software architecture is used to define the overall structure and flow of the system to be developed. The architecture identifies the different components of the system and the communication between the identified components. The design of the architecture of a system can be guided by a set of principles laid down by different architectural styles. The architecture of a system can implement a specific architectural style or a combination of architectural styles that are suitable to solve a particular problem.

The objective of designing an architecture for an NUI supported by Kinect to help facilitate the detection of an FMS is to develop a system that is re-usable and flexible. This allows the system to be used as a base template for future FMS detection. To achieve this objective, a combination of layered and object-oriented architectural styles were used as guiding principles in the design of the NUI architecture. The layered architecture provided the basis for the separation of responsibilities by having three layers each having its own functionality.

The presentation layer focuses on the visualization and interaction of the user with the system. It also retrieves the data from the Kinect to be used for the recognition of movements made by the user. The middle layer has functionality that manages the recognition of the movements performed by the user. The data layer consists of the knowledge base of sub-skills and a log file to store the outcome of a performed movement. The object-oriented architectural style makes use of polymorphism and inheritance to allow for re-usability of code concerning the sub-skills and easy maintenance of individual sub-skills. This fits into the general objective of making the system re-usable and flexible by letting new functionality inherit from already existing functionality. The proposed architecture supports the best practices of the three layered approach by having each layer independent of the other. The use of the object-oriented approach allows for the re-usability of sub-skills making the architecture easily extensible.

The proposed architecture was evaluated by implementing a framework to be used as a template for detecting FMSs using the NUI for Kinect. The purpose of the evaluation was to determine the feasibility of the proposed architecture. The framework should support the requirement of the architecture being re-usable and modifiable. It should be able to perform the functions for which it was intended; that is, to support the detection of FMSs using the Kinect. The proposed framework allows for the inclusion of algorithms used to detect FMSs.

The implementation of the jump instance tested the feasibility of the proposed framework. The jump instance allows for the re-usability of common sub-skills and the independent functionality of different components of the architecture. The jump instance was implemented using the designed three-layered and object-oriented approach. The presentation layer allows the user to view him or herself when performing the action of jumping. A user can also view the feedback of the performed movement by showing whether the sub-skills of the Jump were attempted correctly. The middle layer has functionality to detect the movements of the user and determine the outcome of the jump. The middle layer consists of components to detect if the action performed is a jump, including its associated sub-skills. The outcome of the attempted jump and its associated sub-skills is stored in a log-file.

The developed application needed to be evaluated in terms of accuracy of the system when detecting the performance of a jump and its individual sub-sub-skills. This will be discussed in Chapter 5.

Chapter 5: Evaluation of Jump Detection Prototype

5.1 Introduction

The NUI architecture (Chapter 4) was designed to allow for the implementation of a prototype that detects a particular FMS using the NUI supported by Kinect (Chapter 3). The architecture was evaluated by implementing a framework that can be used as a template for the incorporation of different FMSs. The framework was tested by implementing a prototype which serves as an instance of the framework.

The evaluation (Section 5.2) of the prototype will focus on the detection of the jump FMS in terms of its sub-skills and determine the frequency of success in detecting each of the sub-skills. The experimental design will discuss the procedure followed for the evaluation including the evaluation measure and the instruments used. The evaluation results (Section 5.3) will present the outcome of the tests performed for the jump and each of its sub-skills. The outcome of the sub-skills evaluation will be used to determine the effectiveness of the prototype in providing support for the detection of a legal or illegal jump. The chapter will conclude with a discussion of the effectiveness of the prototype in detecting the performed movement.

5.2 Evaluation

The purpose of the evaluation is to test the accuracy of the prototype in detecting the jump FMS. The prototype should detect the movement of the user as being a legal jump and validate the performance of each of the sub-skills of the jump. The first task of the prototype is to detect the movement performed as being an attempt at a jump. For a jump to be considered legal, the sub-skills (Section 2.4.1) also need to be performed correctly. The second task therefore validates the legality of the jump by detecting the correct outcome of each of the sub-skills. The NUI supported by Kinect can be used to measure the observed sub-skills and provide the information necessary to determine whether the action performed is correct or not (Section 3.4). The selected sub-skills of the jump that can be measured by Kinect are:

1. The knees should be bent before jumping
2. The hands should move back before jumping
3. The arms should be kept straight in the air when jumping
4. The legs should be kept straight when jumping
5. The knees should be bent for a controlled landing.

Three of the sub-skills of the jump could not be detected due to the limitation of the Kinect, namely, the sub-skills of eyes facing forward, landing with the ball of the foot and stabilizing by making one step of the foot in any direction.

5.2.1 Experimental design

The measure chosen to evaluate the prototype is effectiveness. Task success can be used to measure the effectiveness of the developed prototype by accurately detecting an expected outcome, with regard to the following tasks:

- The movement performed is detected as an attempt at a jump;
- For each individual sub-skill, whether the sub-skill is correctly executed or not; and
- Whether the jump performed is a correct jump, by collectively evaluating the sub-skills.

Binary success can be used as a way of measuring task success as the prototype either detects an attempt as correct or not (Tullis & Albert, 2008). A correct attempt at a movement is classified as a positive attempt, while a negative attempt is the classification for an incorrect attempt. When an attempt is detected as being correct, this detection is referred to as a positive detection, while a negative detection is indicated by the detection of an attempt as incorrect. A detection by the prototype consequently results in either a true or a false outcome. Observed frequencies of true and false outcomes are recorded in a contingency table (Table 5.1)

Table 5.1: Contingency Table of Observed Frequencies of Task

	Positive attempt $(\boldsymbol{n_p})$	**Negative attempt** $(\boldsymbol{n_n})$
Positive detection	Observed frequency of true positives (t_p)	Observed frequency of false positives (f_p)
Negative detection	Observed frequency of false negatives (f_n)	Observed frequency of true negatives (t_n)

A true outcome is either a positive detection of a positive attempt (t_p) or a negative detection of a negative attempt (t_n). The closer each of these observed values are to the expected values (n_p and n_n respectively), the better the prototype performs. Other possible outcomes are where the prototype reports a negative detection (f_n) in the case where the attempt is expected to be positive, or where the prototype reports a positive detection (f_p) in the case where the attempt is expected to be negative. For each of the tasks, the desired result of the task evaluation is to have t_p tending towards n_p, t_n tending towards n_n, and both f_n and f_p tending towards 0.

The jump consists of five sub-skills to be measured. The prototype will therefore be subjected to the following accuracy assessments;

- Measuring $n_p = 15$ correct and $n_n = 15$ incorrect attempts for each of the five sub-skills; and
- Measuring $n_p = 15$ correct attempts at a jump FMS incorporating the combination of correctly executed sub-skills, and $n_n = 15$ incorrect attempts at a jump FMS incorporating a variety of combinations of sub-skills, at least one of which is incorrectly executed.

For each of the accuracy assessments, one or more contingency tables will result. From the contingency table, the following measurements can be derived:

The **true positive rate (*tpr*)** represents the probability of the prototype to accurately detect a positive attempt at a movement. High values of true positive rates, that is, the closer the values tend to 1, the better the prototype is able to detect a true outcome of a positive detection.

$$tpr = \frac{t_p}{n_p} \quad \text{Eq.(5.1)}$$

The **true negative rate (*tnr*)** represents the probability of the prototype to accurately detect a negative attempt at a movement. High values of true negative rates, that is, the closer the values tend to 1, the better the prototype is able to detect a true outcome of a negative detection.

$$tnr = \frac{t_n}{n_n} \quad \text{Eq.(5.2)}$$

The **false positive rate (*fpr*)** represents probability of the prototype to detect a false outcome of a movement. Lower values of the false positive rates, that is, values that tend towards 0, indicate that the prototype is less likely to report a false outcome.

$$fpr = \frac{f_p}{n_n} \quad \text{Eq.(5.3)}$$

The **false negative rate (*fnr*)** represents probability of the prototype to detect a false outcome of a movement. Lower values of the false negative rates, that is, values that tend towards 0, indicate that the prototype is less likely to report a false outcome.

$$fnr = \frac{f_n}{n_p} \quad \text{Eq.(5.4)}$$

Accuracy (*acc*) is the measure of how well the prototype is able to detect a true outcome of a performed movement. The higher the accuracy results, the better the prototype performs.

The purpose of the evaluation test was to determine the accuracy of the prototype in detecting the movements performed and determine the effectiveness of the developed prototype. The testing of the prototype aimed to prove that the developed framework could indeed be used to implement a system that detects a particular FMS.

$$acc = \frac{t_p + t_n}{n_p + n_n} \quad \text{Eq.(5.5)}$$

The hardware used for evaluation was a Kinect for windows sensor. The Kinect device was placed 0.9 meters above the floor and the participant stood between 3.0 and 4.0 meters away from the Kinect device.

5.3 Evaluation results

This section will discuss the accuracy results of each of the jump sub-skills detected by the prototype (Section 5.3.1). The accuracy results of the combined sub-skills used to determine a valid and invalid jump will also be discussed (Section 5.3.2).

5.3.1 Sub-skills accuracy detection results

Evaluation of the sub-skills determined the accuracy of the prototype in detecting the jump movement. The evaluation consisted of testing each of the sub-skills to determine the effectiveness of the prototype in detecting correct and incorrect attempts (Appendix A). The independent testing of each of the sub-skills was to determine how accurate the prototype detects a single sub-skill. The outcome of the independent testing is compared to the outcome of the combined testing (Section 5.3.2). This is done to determine if the prototype's performance is affected when detecting a combination of sub-skills (Appendix B). The collected data was recorded in a contingency table to determine the relationship between the positive and negative detections made by the prototype. The success rate of each of the sub-skills in determining the correct and incorrect attempts was determined by analysing the true positive (t_p) and true negative (t_n) outcomes.

Table 5.2: Contingency table of arms move back

	Positive attempt at arms back $(\boldsymbol{n_p = 15})$	**Negative attempt at arms back $(\boldsymbol{n_n = 15})$**
Positive detection	15	0
Negative detection	0	15

For the sub-skill of arms moving back, the results show (Table 5.2) the true positives $(t_p = 15)$ and true negatives$(t_n = 15)$ equal the positive attempts (n_p) and negative attempts (n_n) respectively. Consequently, the observed values of false positives $(f_p = 0)$ and false negatives $(f_n = 0)$ return the desired value of 0. These results show that, for the specific evaluation, the prototype had a hundred percent success rate of detecting correct and incorrect movement of the arms moving back. The calculated results of true positive rate (*tpr = 1.0*) and true negative rate (*tnr = 1.0*) show the prototype's ability to always detect a true outcome of the sub-skill. The false positive rate (*fpr = 0.0*) and false negative rate (*fnr = 0.0*) results show the prototype does not report any false outcomes. This gives the prototype a hundred percent accuracy rate (*acc = 1.0*) in detecting the sub-skill of arms moving back.

Table 5.3: Contingency table of knees bent

	Positive attempt at knees bent $(\boldsymbol{n_p} = \mathbf{15})$	**Negative attempt at knees bent** $(\boldsymbol{n_n} = \mathbf{15})$
Positive detection	15	0
Negative detection	0	15

For the sub-skill of knees bent (Table 5.3), the results show that the true positives $(t_p = 15)$ equal the positive attempts made by the participant. The true negatives $(t_n = 15)$ also report a hundred percent success rate of detecting an incorrect movement of the sub-skill. The prototype reports no false negatives $(f_n = 0)$ or false positives $(f_p = 0)$ for the detected sub-skill. For the evaluation of bent knees, the analysed results show that the prototype is accurate in detecting the correct and incorrect movement of the knees bending. The calculated results of true positive rate (*tpr = 1.0*) and true negative rate (*tnr = 1.0*) shows that the prototype is always able to detect a positive detection and negative detection of the sub-skill. Likewise, the false positive rate (*fpr = 0.0*) and false negative rate (*fnr = 0.0*) result in the prototype only reporting true outcomes. This gives the prototype a hundred percent accuracy rate (*acc = 1.0*) in detecting the sub-skill of knees bending.

Table 5.4: Contingency table of arms stretched

	Positive attempt at arms stretched $(\boldsymbol{n_p} = \mathbf{15})$	**Negative attempt at arms stretched** $(\boldsymbol{n_n} = \mathbf{15})$
Positive detection	12	0
Negative detection	3	15

The displayed results (Table 5.4) show that the true positives $(t_p = 12)$ of the sub-skill of arms stretched are much closer to the positive attempt (n_p). The observed values of the true negatives$(t_n = 15)$ equal the negative attempts (n_n). The prototype reports low values for false negatives $(f_n = 3)$ which are much closer to zero and false positives $(f_p = 0)$ which are equal to the desired values of zero. The result of arms straight shows that the prototype has a high success rate of detecting correct and incorrect movements of the sub-skill. The prototype also reports a low number of false detections as observed from the low number of false positives (f_p) and false negatives (f_n). The calculated results of the true positive rate (*tpr* = 0.8) and true negative rate (*tnr* = 1.0) show that the prototype has a high probability rate in detecting a true outcome of the sub-skill. The false positive rate (*fpr* = 0.2) and false negative rate (*fnr* = 0.0) show that the prototype has a low rate of reporting false outcomes. This gives the prototype a ninety percent accuracy rate (*acc* = 0.9) in detecting the sub-skill of arms straight.

Table 5.5: Contingency table of legs straight

	Positive attempt at legs straight $(\boldsymbol{n_p = 15})$	**Negative attempt at legs straight** $(\boldsymbol{n_n = 15})$
Positive detection	13	2
Negative detection	2	13

The results of the sub-skill of legs straight (Table 5.5) show, for the true positives $(t_p = 13)$ and true negatives $(t_n = 13)$, the observed value is much closer to the positive attempt (n_p) and (n_n). The prototype reports low values for false negatives $(f_n = 2)$ and false positives$(f_p = 2)$. The results indicate that, for the sub-skill of arms straight, the prototype has a high success rate of detecting correct movements and incorrect movements. The calculated results of the true positive rate (*tpr* = 0.87) and true negative rate (*tnr* = 0.87) show that the prototype has a high probability of detecting a true outcome of the sub-skill. The false positive rate (*fpr* = 0.13) and false negative rate (*fnr* = 0.13) result in the prototype reporting a low number of false outcomes. This gives the prototype an eighty-seven percent accuracy rate (*acc* = 0.87) in detecting the sub-skill of legs being straight.

Table 5.6: Contingency table of controlled landing

	Positive attempt at controlled landing $(n_p = 15)$	**Negative attempt at controlled landing $(n_n = 15)$**
Positive detection	12	0
Negative detection	3	15

The results for the sub-skill of controlled landing (Table 5.6) show that the true positives $(t_p = 12)$ tend to the maximum positive attempt (n_p) and the observed value of the true negatives$(t_n = 15)$ is equal to the negative attempts (n_n). The observed value of the false negatives $(f_n = 3)$ shows a much lower value that is closer to zero and the observed false positives $(f_p = 0)$ are equal to the expected value of zero. This shows that, for the specific evaluation of controlled landing, the prototype has a high success rate of detecting a correct and an incorrect movement of the sub-skill. The prototype also reports a low outcome of false detections of the sub-skill. The calculated results of true positive rate (*tpr* = 0.8) and true negative rate (*tnr* = 1.0) show that the prototype has a high probability rate of detecting a true outcome of the sub-skill. The false positive rate (*fpr* = 0.0) and false negative rate (*fnr* = 0.2) results show a very low probability rate of false outcome of the sub-skill. The prototype has a ninety percent accuracy rate (*acc* = 0.9) in detecting the sub-skill of knees bending.

Evaluation of the prototype reveals high levels of accuracy in the independent detection of each of the sub-skills of the jump FMS (Table 5.7), with the lowest level of accuracy being that for detection of legs straight (*acc = 0.87*)

Table 5.7: Accuracy results

Sub-Skills	Accuracy
Legs Straight	0.87
Arms Straight	0.90
Controlled Land	0.90
Bent Knees	1.00
Arms Back	1.00

5.3.2 Jump accuracy detection

The second test of the prototype was to measure the correct attempt at a jump incorporating the combination of correctly executed sub-skills. A valid attempt at a jump involves the correct identification of the combination of each of the sub-skills (Table 5.8). For this experiment, all sub-skills for all 15 jumps were performed correctly.

Table 5.8: Correct attempts at a jump FMS

Jump number	Arms Back	Arms Straight	Bent Knees	Controlled landing	Legs Straight	Valid Jump
1	🟩	🟩	🟩	🟩	🟩	🟩
2	🟩	🟩	🟩	🟩	🟩	🟩
3	🟩	🟩	🟩	🟩	🟩	🟩
4	🟩	🟩	🟩	🟩	🟩	🟩
5	🟩	🟩	🟩	🟩	🟩	🟩
6	🟩	🟩	🟩	🟩	🟩	🟩
7	🟩	🟩	🟩	🟩	🟩	🟩
8	🟩	🟩 ✗	🟩	🟩	🟩 ✗	🟩 ✗
9	🟩	🟩	🟩	🟩	🟩	🟩
10	🟩	🟩 ✗	🟩	🟩	🟩	🟩 ✗
11	🟩	🟩	🟩	🟩	🟩	🟩
12	🟩	🟩	🟩	🟩	🟩	🟩
13	🟩	🟩	🟩	🟩	🟩	🟩
14	🟩	🟩	🟩	🟩	🟩	🟩
15	🟩	🟩 ✗	🟩	🟩	🟩 ✗	🟩 ✗

🟩 Correct attempt ✗ Incorrect detection

The total outcome of each of the sub-skills detected for each jump appears in Table 5.8. A valid attempt at a jump involves the positive detection of the combination of each of the sub-skill. That is, for a single jump, each of the sub-skills should show a true positive outcome.

An invalid attempt at a jump involves the correct identification of an incorrect performance of one or more of each of the sub-skills. The results show that the prototype detects 12 of 15 correct attempts at a jump, as being correct.

Table 5.9: Incorrect attempts at a jump

Jump number	Arms Back	Arms Straight	Bent Knees	Controlled landing	Legs Straight	Invalid jump
1	🟩	🟥	🟩	🟩	🟥	🟩
2	🟩	🟥	🟩	🟩	🟥	🟩
3	🟩	🟥	🟩	🟩	🟥	🟩
4	🟩	🟥	🟩	🟩	🟥	🟩
5	🟩	🟩	🟩	🟥	🟩 ✗	🟩
6	🟩	🟩	🟩	🟥	🟩 ✗	🟩
7	🟩	🟩	🟩	🟥	🟩	🟩
8	🟩	🟩 ✗	🟩	🟥	🟩 ✗	🟩
9	🟩	🟩	🟩	🟥	🟩 ✗	🟩
10	🟩	🟥	🟩	🟥	🟩	🟩
11	🟥	🟥	🟩	🟥	🟩	🟩
12	🟥	🟥	🟩	🟥	🟩	🟩
13	🟥	🟥	🟩	🟩	🟩	🟩
14	🟥	🟥	🟩	🟩	🟩 ✗	🟩
15	🟥	🟥	🟩	🟥	🟩	🟩

🟩 Correct attempt 🟥 Incorrect attempt ✗ Incorrect detection

Table 5.9 shows the outcome of 15 incorrect attempts at jump. The prototype accurately detected all 15 attempts as incorrect attempts. The incorrect attempts at a jump are determined by a combination of correct and incorrect sub-skill attempts. The only sub-skill to be tested with all positive attempts is bent knees. This is because with the detection of a combination of sub-skills, bending of knees is needed for a jump to be registered. The combination of sub-skills when detecting an incorrect attempt at a jump indicates an increased error rate for the sub-skills of legs straight. The prototype has difficulty detecting straight legs when detecting a combination of sub-skills of an incorrect attempt at a jump. The outcome of the correct and incorrect attempts at a jump can be recorded in a contingency table (Table 5.10) and analysed to determine the accuracy of the jump detection.

Table 5.10: Contingency table of attempts at jump

	Positive attempt at jumping $(\boldsymbol{n_p = 15})$	**Negative attempt at jumping** $(\boldsymbol{n_n = 15})$
Positive detection	12	0
Negative detection	3	15

The results of the attempt at jumping (Table 5.10) show, for the true positives$(t_p = 12)$, the observed value is much closer to the positive attempt (n_p). The observed values of the true negatives$(t_n = 15)$ are equal to the expected value of zero. The prototype reports low values for false negatives ($f_n = 3$) and false positives$(f_p = 0)$. The calculated results of true positive rate (*tpr* = 0.8) show that the prototype has a high probability of detecting a true outcome of a correct attempt at a jump. The true negative rate (*tnr* = 1.0) shows that the prototype has a high probability of detecting an incorrect attempt at a jump. The results show a low false positive rate (*fpr* = 0.0) and false negative rate (*fnr* = 0.2). This gives the prototype a ninety percent accuracy rate (*acc* = 0.9) in detecting a correct and an incorrect attempt at a jump.

The outcome of the combined sub-skills detections, show a close relationship with the individual sub-skill detections for positive and negative detections. The relationship can be used to confirm the ability of the prototype to effectively detect the correct and incorrect attempt of an FMS.

5.4 Conclusion

The objective of this chapter was to determine the effectiveness of the implemented prototype using the proposed architecture and framework. An instance of the jump was implemented to determine whether the proposed framework would support its implementation. The implemented prototype was then evaluated to determine its accuracy in detecting the correct and incorrect performance of the sub-skills associated with the jump. Each sub-skill of the jump was tested on the rate at which it is able to report the correct values of the performed movement. The calculated rate determined the true positive rate of the sub-skill evaluated. The evaluation also determined the true negative rate which reported the rate at which the prototype determined incorrect movement when the movement was indeed incorrect. The false positives and false negative rates revealed the false outcomes reported by the prototype for each of the sub-skills. The positive and negative detection of an attempt at a jump was evaluated by detecting the sub-skills simultaneously.

The results of the evaluation showed that the proposed framework supports the implementation of the prototype in that the prototype proved to be effective in the detection of the sub-skills associated with the jump. The prototype evaluation showed high accuracy rates for the detection of correct and incorrect performances of the sub-skills. The high accuracy rate was achieved by having high positive and negative rates for the detection of the sub-skills. The low rate of false positive and false negative results showed that the prototype is less likely to detect a false outcome for a performed sub-skill. The prototype also showed a high accuracy rate for the combined sub-skills of a positive and negative attempt at a jump. This result showed that the prototype is able to detect a correct and an incorrect attempt at a jump.

The effectiveness of the developed prototype shows that an FMS can be implemented using the proposed framework. The implementation of the framework showed that it can be used as a template for the implementation of FMSs using the proposed architecture (Chapter 4). The results indicate that the proposed architecture can be used as a foundational blueprint to be used to implement future FMSs using the NUI supported by Kinect.

Chapter 6: Conclusions

6.1 Introduction

The learning of an FMS takes time and one needs to focus on the correct performance of the sub-skills of an FMS in order to achieve proficiency in a particular FMS. Tools and equipment can be used to help facilitate the detection of FMSs and the associated sub-skills. Technologies using NUIs can be used to monitor movements made by a user. One of these technologies is the NUI supported by Kinect and it can be used to help facilitate the detection of FMSs by monitoring the movements of the user and detecting whether the FMS is performed correctly as whole and in terms of its sub-skills.

The developed application using the Kinect can be used to give feedback on the performance of an FMS and allow the user to identify the sub-skills they are not able to perform correctly. This research aimed to propose an effective NUI architecture using the Kinect to facilitate the detection of FMSs. A framework was implemented from the proposed architecture, thereby demonstrating the feasibility of the proposed architecture. The framework was evaluated by incorporating an instance of the jump FMS as a case study to determine the instance's effectiveness in detecting an FMS.

The objectives of the research and the research questions will be reviewed to determine whether they have been achieved. The objectives of the research were identified (Section 1.4) as follows:

1. To investigate and describe the types of fundamental movement skills and the challenges in teaching them (Section 6.2);
2. To propose an NUI architecture to be used in the detection of fundamental movement skills using the features of the Kinect (Section 6.4);
3. To apply and evaluate the proposed architecture (Section 6.5).

These research questions were identified (Section 1.5) as follows:

1. What are fundamental movement skills?
2. What are the challenges of teaching and learning fundamental movement skills?
3. How can Kinect support a natural user interface to detect fundamental movement skills?
4. How is the Kinect used?
5. How can a natural user interface architecture be designed?
6. How can a prototype of a Kinect be implemented by applying the architecture?
7. How effective is the prototype in the detection of fundamental movement skills?

The objective of this chapter is to give a review of the research and reflect on the research contributions including the implications of the research (Section 6.4). The chapter also discusses the limitations (Section 6.5) encountered during the course of the research, as well as recommendations made for future research (Section 6.6).

6.2 Types of FMSs and the challenges in teaching them

The research questions to be addressed in this section are:

RQ1 - What are fundamental movement skills?

RQ2 - What are the challenges of teaching and learning fundamental movement skills?

This study undertook an in-depth investigation of FMSs and the challenges in teaching them (Chapter 2). FMSs are described as movements with specific observable patterns. These are divided into three categories, namely locomotor skills involving the movement of the body from one place to another; non-locomotor skills involving movement in a stationary position and manipulative skills involving object manipulations (Section 2.4). The different FMSs each have specific sub-skills (observable patterns) that need to be performed correctly for an FMS to be correct. The identified sub-skills of each of the FMSs were categorised into groups consisting of the head (sub-skills concerning head movement), arms (sub-skills concerning arm movement), knees (sub-skills concerning knee movement) and feet (sub-skills concerning feet movement). This was done to identify the common sub-skills belonging to the FMSs. The categories would also be used as a criterion to determine the technology most suitable for the tracking and detecting of an FMS.

To achieve proficiency of FMSs, children require teaching and instruction in the correct performance of an FMS. Children need to be given opportunities for practice in improving an FMS performance. The challenge faced by teachers is finding enough time and practice opportunities to implement programmes to teach correct movement skills. Free play provides opportunities to learn FMSs but children do not learn the correct way of performing these skills during free play. Children need to be observed by the teachers and given instruction and feedback on how to perform correct FMSs. The teachers need to be provided with tools to help them teach FMSs to children rather than letting the children participate in free play. By incorporating children's love of games and free play, teachers could be supplied with tools that can be used to teach FMSs correctly while exciting the children also.

6.3 NUI supported by Kinect

Research questions addressed:

RQ3 - What are the features of the natural user interface supported by Kinect?

RQ4 - How is the Kinect used?

Chapter 3 presented a discussion on NUIs. Due to the lack of tools that could be used to learn FMSs, an investigation into NUIs could be used to determine appropriate devices that could be as tools in the context of learning FMSs. NUIs allow users to interact with devices using the natural gestures. This research focused on the NUI supported by Kinect which allows the user to interact with the device using his/her whole body. This is achieved by the device's ability to be able to detect and track full body movements made by a user. The NUI supported by Kinect tracks a user's movement by monitoring the skeleton joints of the user. By implementing gesture recognition algorithms, the Kinect can be used to track skeleton data to determine the outcome of the movement performed by the user. The Kinect is able to recognize up to 20 tracked joints of a human skeleton. Using the categorised sub-skills information of the FMSs and the skeleton joints that can be tracked by the Kinect, the FMS that could be detected by the Kinect were identified. The selected FMS to be implemented as a case study was identified by choosing the locomotor skill that had the most sub-skills tracked by the Kinect.

6.4 NUI architecture design for detecting FMSs using the Kinect

Research question addressed:

RQ5 - How can a natural user interface architecture be designed?

An NUI architecture supported by Kinect for detecting FMSs was proposed (Figure 6.1). The requirements of the design of the architecture support a plug and play system to make additions of future FMSs easier. The architecture allows for the separation of functionalities making it loosely coupled. This allows for changes to be made to one part of the system without affecting the whole system. The design of the proposed architecture consists of a combination of the three-layer and object-oriented architectural styles. The three layer architectural style satisfies the requirement of separation of responsibilities by having each layer focusing on specific functionality. It also satisfies the plug and play ability as future FMSs can be added to a specific layer of the architecture. The object-oriented architectural style satisfies the requirement of the plug and play ability by re-use of functionality through the addition of FMSs. The three layers of the architecture design are the presentation layer, middle layer and data layer.

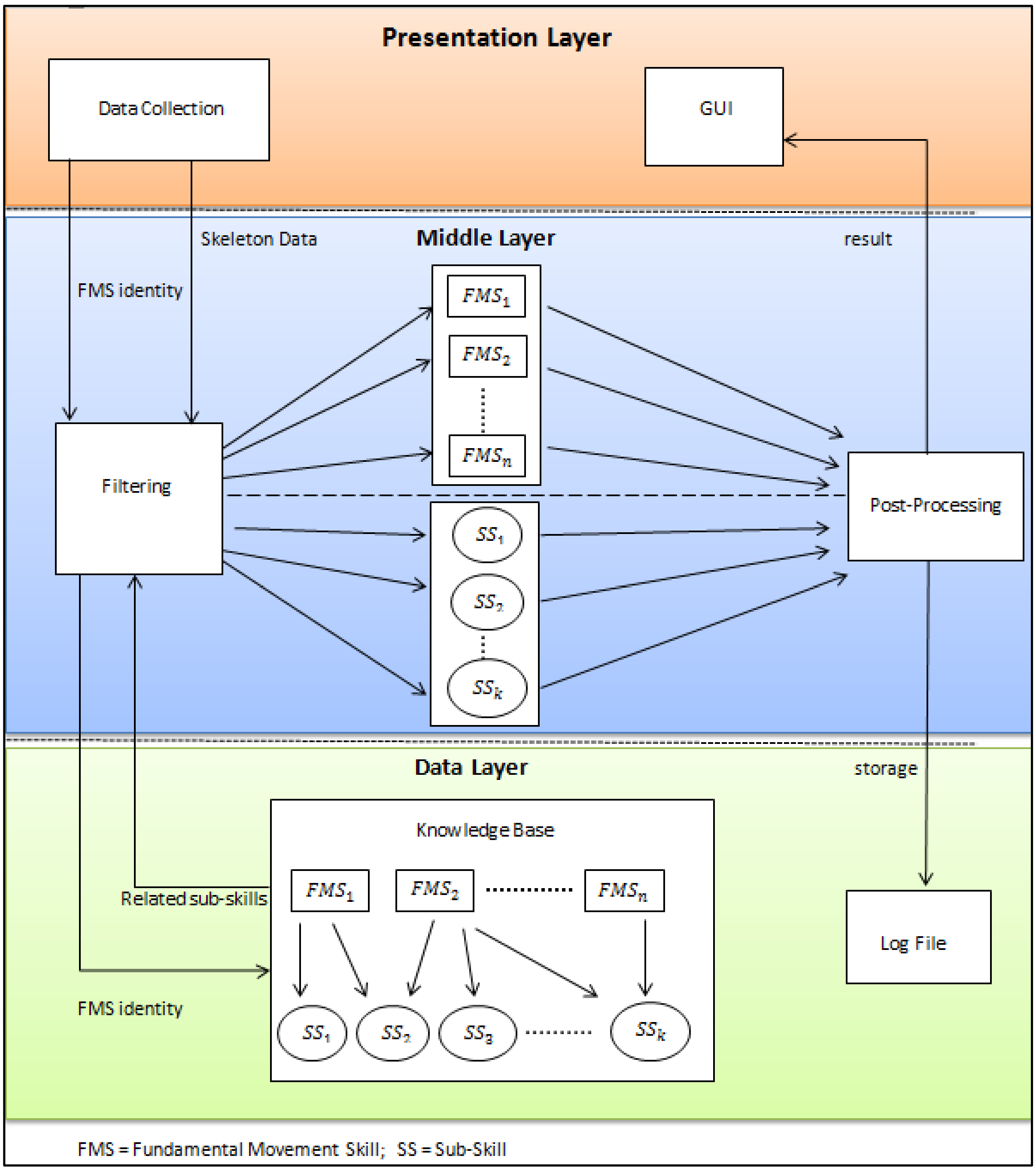


Figure 6.1: Proposed architecture of FMSs

The presentation layer is responsible for the collection of data from the Kinect and extracting the relevant data to be used for the detection of movements. The data collected from the Kinect is in the form of depth and skeletal data. The skeletal data consists of tracked skeleton joints of the user being detected by the Kinect. The layer is also responsible for the display of the performed movement's outcome and visualization of how the user is performing the movement. The retrieved data from the Kinect is communicated to the middle layer for processing.

The middle layer receives data from the presentation layer. It implements the gesture recognition algorithms used to detect the FMS performed by the user. It consists of four components, namely; filtering, FMS, sub-skills and post-processing components. The filtering component is responsible for identifying the FMS to be detected including its associated sub-skills. The FMS component is responsible for the management and maintenance of the FMS. The sub-skill component determines the outcome of the sub-skills associated with the FMS being detected. The post-processing component determines the outcome of the FMS. The results of the gesture detection outcome are communicated to the presentation layer for display and the data layer for storage.

The data layer receives information from the middle layer. The data received is used to retrieve the relationship between the identified FMS and its associated sub-skills. The data layer allows the common sub-skills to be re-used by different FMSs. The data layer also stores the outcome of the performed FMS in a log file. The stored outcome of an FMS performance can be reviewed and feedback can be given based on the performance of the sub-skills.

6.5 Prototype

The research questions addressed in this section are:

> RQ6 - How can a prototype of a Kinect be implemented by applying the architecture?
>
> RQ7 - How effective is the prototype in the detection of fundamental movement skills?

A prototype was developed by incorporating an FMS to the implemented framework (Section 6.5.1) and the prototype (Section 6.5.2) was evaluated for effectiveness.

6.5.1 NUI framework

The implementation of the framework served as a template to be used for the incorporation of an instance to detect an FMS. The framework provided generic functionality that could be used to implement an FMS. It was implemented using the three-layer and object-oriented architectural styles.

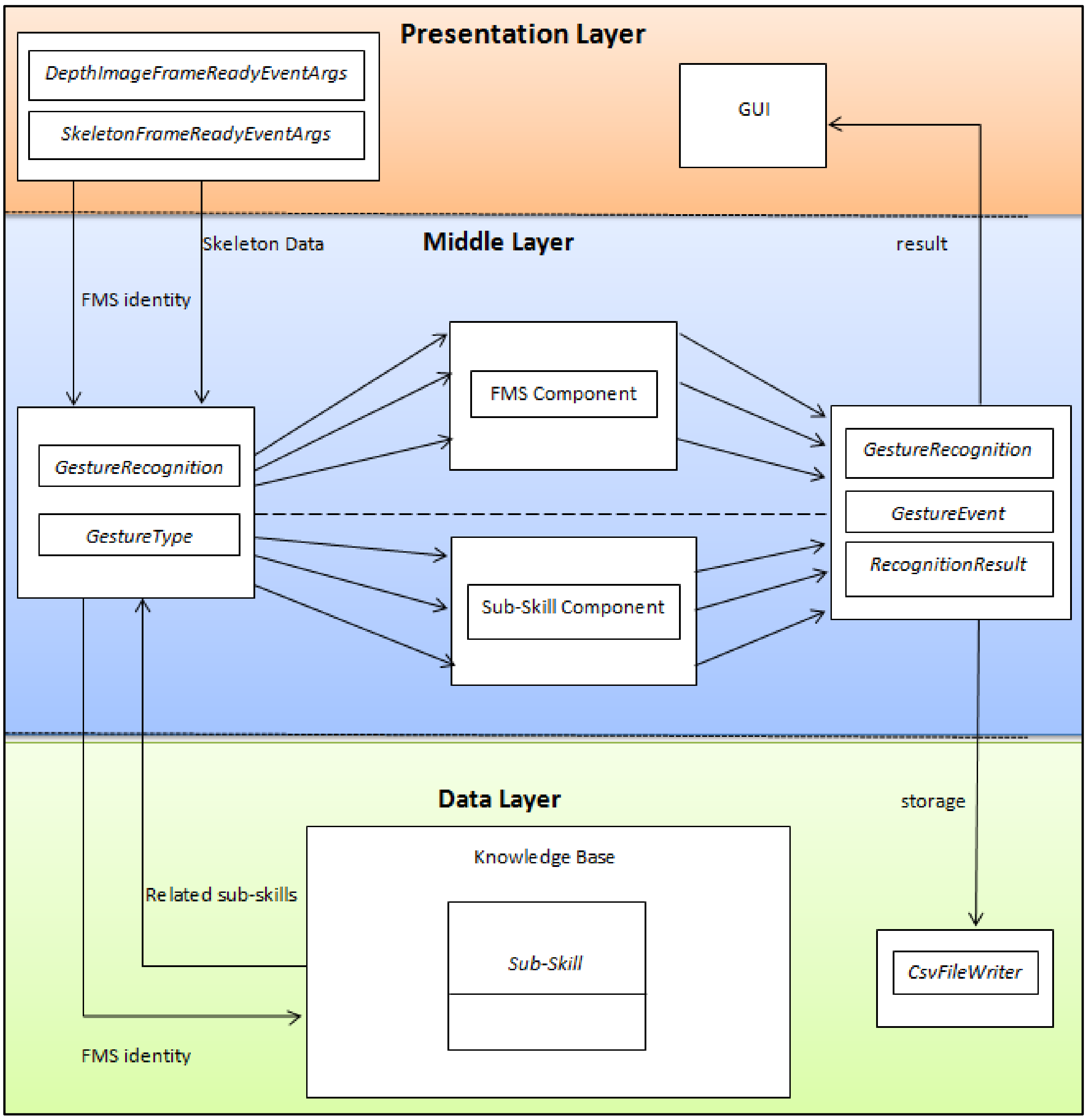


Figure 6.2: Proposed framework

The presentation layer has functionality to retrieve information from the Kinect and display information on the user interface. It is implemented using the Windows Presentation Foundation (WPF) control library that can be used to connect to the Kinect and display the outcome on the user interface screen. The middle layer consists of classes to manage functionality related to gesture detection of the performed movement. It is implemented using a gesture detection library that has functionality to recognize detected FMS types, detected identified FMSs and to determine the outcome of the identified FMS to be performed. The data layer consists of a base class for the FMS sub-skill functionality and data storage functionality. It consists of a comma separated values (CSV) file writer that stores the outcome of the performed FMS in a CSV file. The layer also consists of the main base sub-skill class and added sub-skills can inherit common functionality from the base class.

6.5.2 Development of prototype

The jump FMS was implemented as a case study to test the framework's robustness and adherence to the architecture. The prototype was developed using the generic framework classes and implemented using specific functionality related to the jump. The three-layered approach corresponding to the architecture was followed. The first layer is the presentation layer and it involves the retrieval of depth and skeleton data used to detect the attempted jump performed by the user. The collected data is communicated to the middle layer by sending the identity of the FMS as a jump and the tracked joints of the user. The second layer is the middle layer and it involves the implementation of the FMS class for the jump gesture recognition. The middle layer also processes the sub-skill classes corresponding to the jump. The algorithmic approach is used to implement the algorithms to detect the jump FMS and its sub-skills. The generic gesture detection class is used to identify the outcome of the performed movement as a jump. The data layer implements the jump's associated sub-skills that inherits common functionality from the base sub-skill class. The outcome of the jump detection from the middle layer is written to a CSV file and can be reviewed for feedback.

6.5.3 Evaluation of prototype

The prototype was evaluated in terms of the sub-skills of the jump FMS. The purpose of the evaluation was to test the effectiveness of the implemented prototype in detecting the correct performance of the jump FMS. The criterion for a correct performance of a jump involves the correct detection of the sub-skills of the jump.

The evaluation consisted of the participant performing three tasks. The tasks consisted of confirming the movement as an attempt at a jump, attempting a correct and an incorrect execution of each of the sub-skills of the jump and the correct attempt at a jump by evaluating the combined sub-skills. The results of the outcomes were recorded in contingency tables and analysed to determine the effectiveness of the prototype in detecting a correct performance of the jump FMS.

The results indicate high accuracy rates for the correct and incorrect detection of each of the sub-skills of the jump. The prototype also scored highly on all but one sub-skill of the combined detection. The sub-skills of legs straight have a higher false negative rate when combined with other sub-skills for an incorrect attempt at a jump. The prototype had a high accuracy for detecting the correct and incorrect detection for an attempt at a jump.

6.6 Contributions

The outcome from the research objectives has led to both theoretical (Section 6.6.1) and practical contributions (Section 6.6.2) of the research.

6.6.1 Theoretical contributions

This research has identified the NUI supported by Kinect as a tool that can be used to detect FMSs. A major theoretical contribution of this study is the NUI architecture (Figure 6.1) which is discussed in section 6.4. As a further theoretical contribution, an experimental design was proposed to evaluate the effectiveness of the architecture.

The experimental design involves the process of evaluating the architecture. The evaluation of the architecture involves the implementation of the framework. The framework serves to prove that the designed architecture satisfies the requirements of being scalable. The evaluation also includes the creation of an instance of an FMS (prototype) using the framework. The effectiveness of the prototype was determined by evaluating the instance.

The design of the experiment involved the evaluation of the prototype for the following three tasks:

- The movement performed is detected as an attempt at an FMS;
- For each individual sub-skill, detect whether the sub-skill is correctly executed or not; and
- Whether the FMS performed is a correct attempt, by collectively evaluating the sub-skills.

Table 6.1: Contingency Table of Observed Frequencies of Task

	Positive attempt $(\boldsymbol{n_p})$	**Negative attempt** $(\boldsymbol{n_n})$
Positive detection	Observed frequency of true positives (t_p)	Observed frequency of false positive (f_p)
Negative detection	Observed frequency of false negative (f_n)	Observed frequency of true negative (t_n)

The detection of each sub-skill of an FMS is measured and recorded in a contingency table (Table 6.1). The prototype is subject to the following accuracy assessments of each of the sub-skills and the combined sub-skills of an attempt at a jump:

- Measuring n_p = total set of correct attempts and n_n = total set of incorrect attempts at the sub-skills; and
- Measuring n_p = total set of correct attempts at an FMS incorporating the combination of correctly executed sub-skills, and n_n = total set of incorrect attempts at an FMS incorporating a variety of combinations of sub-skills, at least one of which is incorrectly executed.

The outcome of each of the accuracy assessments results in one or more contingency tables to be used to derive the following measurements:

$$True\ positive\ rate\ (tpr) = \frac{t_p}{n_p} \qquad \text{Eq.(5.1)}$$

$$True\ negative\ rate\ (tnr) = \frac{t_n}{n_n} \qquad \text{Eq.(5.2)}$$

$$False\ positive\ rate\ (tpr) = \frac{f_p}{n_n} \qquad \text{Eq.(5.3)}$$

$$False\ negative\ rate\ (tpr) = \frac{f_n}{n_p} \qquad \text{Eq.(5.4)}$$

$$Accuracy\ (acc) = \frac{t_p + t_n}{n_p + n_n} \qquad \text{Eq.(5.5)}$$

The calculated values are used to determine the effectiveness of the prototype in detecting the true outcome of a performed FMS. The higher the rates of true positive and true negative rates determine the likelihood of the prototype in reporting a true outcome. The lower rates of the false positive and false negative rates represent the likelihood of the prototype in reporting a false outcome. High accuracy shows that the prototype is effective in detecting true outcomes of the performed FMS.

6.6.2 Practical contribution

The practical contribution consisted of the implementation of a framework. The framework (Figure 6.2) implementation served as an evaluation of the proposed architecture. The framework was evaluated by incorporating an instance of an FMS which served as the prototype to detect the jump FMS. The developed prototype was evaluated to determine its effectiveness in detecting an FMS, using the proposed experimental design (Section 5.2.1). The accuracy of the sub-skills associated with the jump was determined. The combined sub-skills measurements were used to determine the accuracy of the prototype to detect a correct and an incorrect attempt at a jump (Table 6.2).

Table 6.2: Accuracy results of sub-skills and jump FMS

Sub-Skills	**Accuracy**
Legs Straight	0.87
Arms Straight	0.90
Controlled Land	0.90
Bent Knees	1.00
Arms Back	1.00
Jump FMS	0.90

The evaluation of the prototype served to prove that the proposed architecture could be used to implement a NUI supported by Kinect to facilitate the detection of an FMS and it is accurate in detecting the correct outcome. The evaluation of the architecture consisted of three levels:

- The architecture was applied in the development of a framework;
- The framework was evaluated by incorporating an instance to detect a selected FMS; and

- The instance was evaluated to accurately detect the selected FMS.

6.7 Implications of research

The proposal of an NUI architecture to facilitate the detection of FMSs using the Kinect, allows developers to implement algorithms to detect an FMS and re-use the relevant common sub-skills they have with other FMSs. The design of the proposed architecture allows for flexibility and maintenance of the system by making use of the separation of functionality among the three layers. This allows developers to add future FMSs or make changes to one layer without affecting the other layers.

The evaluation of the architecture was achieved by implementing a framework and incorporating an instance of an FMS in the framework. The results of the evaluated prototype showed that the prototype was able to accurately detect the correct and an incorrect performance of an FMS. The accuracy of the prototype in detecting a correct and incorrect performance of an FMS means that it could be used as a tool to assess FMSs performed by children and thereby be used in the learning of FMSs.

6.8 Limitations of research

The research was affected by several limitations. One of the limitations faced was that the Kinect can only detect movements that happen within a specified viewable area. This means that FMSs like galloping which cause the user to move out of the view area cannot be detected by the Kinect. Regarding the jump FMS, the Kinect could not detect the sub-skill of landing on the balls of the feet because it is not very accurate in detecting small changes between joints. The Kinect could also not detect the sub-skills of the eyes facing forward because it cannot obtain information about the orientation of the head (Section 3.3).

The Kinect has a minimum height limitation of 1.0 m, which means that children below this height cannot be tracked by the Kinect. FMSs are taught to children between the ages of two and seven years of age. The average height of a two-year-old is 0.89 m and 1.0 m for a four-year-old. Only children aged four to seven could thus generally be detected by the Kinect.

A single participant was used in the evaluation of the prototype because it was used to show proof of concept that the NUI for kinect can be used to detect movements performed by a user.

6.9 Recommendations for future research

FMSs are developed over time and children need to be given a sufficient amount of time and instruction to achieve proficiency in the performance of FMSs. The envisaged future research recommendation would involve the development of a complete FMS system to be used in a field study to determine the impact on the development of FMSs using the system. The research would also observe the impact on how the system as a tool can help teachers structure and teach FMSs to children in the foundation phase of development. The research can also be extended to develop a game that can be used to teach children FMSs.

From section 6.8 it is clear that the Kinect has certain limitations (accuracy of detection and height of participant). Future research could investigate addressing this by incorporating other NUI devices as part of the solution.

6.10 Summary

FMS proficiency is important for the development of movement skills in children. Proficiency in FMSs allows children to participate in more specialized forms of movements and sporting activities. Proficiency of an FMS is identified by the correct performance of sub-skills associated with the FMS. To become proficient in the performance of an FMS, children need to be given appropriate instructions and practice opportunities. The limitation is that teachers do not have enough time to teach the correct performance of FMSs, and resort to general free play. Teachers need to be provided with tools that can assist children in the learning of FMSs. A possible solution is to use technological tools using NUI to help facilitate the learning of FMSs in children.

The goal of this research was to propose an effective NUI architecture using the Kinect to facilitate the detection of FMSs. The proposed architecture defines the essential guidelines to develop a flexible and re-usable system. The architecture’s plug and play ability allows developers to add FMSs by specifying functionality that is relevant to the FMS and re-using available functionality. The effectiveness of the proposed architecture was determined by the implementation of a framework and evaluation of an FMS as a case study to detect the correct performance of the movement skill. The results of the evaluation show that the proposed architecture can be used to develop an NUI supported by Kinect that can be used to facilitate the detection of an FMS. The proposed architecture can thereby be used to provide a foundation for a system that can be used to facilitate the learning of an FMS using the NUI for Kinect.

References

Achper Victorian Branch. (2008). *Fundamental motor skills module*. Retrieved October 1, 2013, from https://www.eduweb.vic.gov.au/edulibrary/public/teachlearn/student/phasefmsmod.pdf.

Adams, J. A. (1971). A closed-loop theory of motor learning. *Journal of Motor Behavior, 3*(2), 111–150. doi:10.1080/00222895.1971.10734898.

Alatalo, P., Jarvenojal, J., Karvonen, J., Keronen, A., & Kuvaja, P. (2002). Mobile application architectures. *Product Focused Software Process Improvement* (pp. 574–575). Springer Berlin Heidelberg. doi:10.1007/3-540-36209-6_47.

Alfredo, P., Tore, D., & Discepolo, T. (2013). Natural user interfaces as a powerful tool for courseware design in Physical Education. *Journal of e-Learning and Knowledge Society*, *9*(May), 105–114.

Apple iOS. (2013). *Siri*. Retrieved May 5, 2013, from http://www.apple.com/ios/siri/.

Atchley, J., Neish, J., Payne, P., Stoddard, C., Watts, R., & Watts, L. (2012). F*undamental movement phase*. Retrieved April 23, 2013, from http://www.pgpedia.com/f/fundamental-movement-phase.

Barnett, L., Morgan, P., Van Beurden, E., & Beard, J. (2008). Perceived sports competence mediates the relationship between childhood motor skill proficiency and adolescent physical activity and fitness: a longitudinal assessment. *International Journal of Behavioral Nutrition and Physical Activity*, *5*, 40. doi:10.1186/1479-5868-5-40.

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice* (2nd Ed :19–24). Addison-Wesley Professional.

Bianco, P., Kotermansk, R., & Merson, P. (2007). *Evaluating a service-oriented architecture*. Retrieved August 5, 2014, from http://www.sei.cmu.edu/reports/07tr015.pdf.

Blake, J. (2012). *Natural User Interfaces in .Net* (MEAP : 5–7). Manning Publications.

Booch, G. (1998). O*bject-oriented analysis and design with applications :* 25–78. Addison Wesley Longman, Inc.

Booth, M. L., Okely, T., McLellan, L., Phongsavan, P., Macaskill, P., Patterson, J., Wright, J., et al. (1999, June). Mastery of fundamental motor skills among New South Wales school students: prevalence and sociodemographic distribution. *Journal of* *science and medicine in sport / Sports Medicine Australia*. Retrieved August 8, 2013, from http://www.ncbi.nlm.nih.gov/pubmed/10476973.

Boulos, M. N. K., Blanchard, B. J., Walker, C., Montero, J., Tripathy, A., & Gutierrez-Osuna, R. (2011). Web GIS in practice X: A Microsoft Kinect natural user interface for Google Earth navigation. *International journal of health geographics*, *10*(1), 45. doi:10.1186/1476-072X-10-45.

Bowman, D. A., McMahan, R. P., & Ragan, E. D. (2012). Questioning naturalism in 3D user interfaces. *communications of the ACM*, *55*(9), 78–88. doi:10.1145/2330667.2330687.

Bowman, D., Kruijff, E., LaViola, J., & Poupyrev, I. (2005). *3D user interfaces: Theory and practice*. Addison-Wesley. Boston: Addison-Wesley. Retrieved August 20, 2014, from http://ptgmedia.pearsoncmg.com/images/9780201758672/samplepages/0201758679.pdf.

Breivold, H. P. (2011). *Software architecture evolution through evolvability analysis*. Retrieved July 7, 2014, from http://www.diva-portal.org/smash/get/diva2:446143/FULLTEXT01.pdf.

Breslin, C. , Morton, J. , & Rudisill, M. (2008). Implementing a physical activity curriculum into the school day: Helping early childhood teachers meet the challenge. *Early Childhood Education Journal,*, *35*, 429–437.

Butow, E. (2007). *User interface design for mere mortals*. (M. Hernandez, Ed.) (p. 28). Pearson Education, Inc.

Chapinal Cervantes, J., Vela, F. L. G., & RodrÍguez, P. P. (2012). Natural interaction techniques using kinect. *Proceedings of the 13th International Conference on Interacción Persona-Ordenador - INTERACCION '12*. New York, New York, USA: ACM Press. doi:10.1145/2379636.2379650.

Clark, J. E. (2007). On the problem of motor skill development. *Journal of Physical Education, Recreation & Dance*, *78*(5), 39–44.

Decorby, K., Halas, J., Dixon, S., Wintrup, L., & Janzen, H. (2005). Class Teachers and the Challenges of Delivering Quality Physical Education. *The Journal of Educational Research*, 98(4), 208–221.

Denning, P. . (1997). A New Social Contract for Research. *Communications of the ACM*, *40*(2), 132–134.

Department of Education. (2009). *Fundamental motor skills: A manual for classroom teachers*. Victoria. Retrieved March 12, 2013, from https://www.eduweb.vic.gov.au/edulibrary/public/teachlearn/student/fmsteachermanual09.pdf.

Department of Education. (2013). *Fundamental movement skills: Book 1 - Learning, teaching and assessment*. Retrieved April 24, 2013, from http://det.wa.edu.au/stepsresources/detcms/navigation/fundamental-movement-skills/.

Educause. (2008). *7 Things You Should Know About Multi-Touch Interfaces*. Retrieved September 30, 2013, from http://net.educause.edu/ir/library/pdf/eli7037.pdf.

EduCLIME. (2013). *Motor skills are essential to learning*! Retrieved August 17, 2013, from http://www.educlime.com/contactus.html.

Ennis, C. D. (2011). Physical Education Curriculum Priorities : Evidence for Education and Skillfulness Physical Education Curriculum Priorities : Evidence for Education and Skillfulness. *Quest*, 63(1), 5–18.

Fielding, T. (2000). *Architectural styles and the design of network-based software architectures*. Doctoral dissertation, University of California. Retrieved April 16, 2014, from http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Fredericks, C. R., Kokot, S. J., & Krog, S. (2006). Using a developmental movement programme to enhance academic skills in Grade 1 learners. *South African Journal for Research in Sport, Physical Education and Recreation*. Retrieved March 24, 2013, from http://reference.sabinet.co.za/webx/access/electronic_journals/sport/sport_v28_n1_a3.pdf.

Gagen, L. M., & Getchell, N. (2006). Using “Constraints” to Design Developmentally Appropriate Movement Activities for Early Childhood Education. *Early Childhood Education Journal*, *34*(3), 227–232. doi:10.1007/s10643-006-0135-6.

Gallahue, D., & Cleland-Donnelly, F. (2007). *Developmental physical education for all children.* (5th ed.). Champaign, IL: Human Kinetics.

Gallahue, D. L., & Donnelly, F. C. (2003). *Developmental physical education for all children* (4th ed.). Human Kinetics.

Gallahue, D. L., Ozmun, J. C., & Goodway, J. D. (2012). *Understanding Motor Development: Infants, Children, Adolescents, Adults* (7th Ed : 47–62). McGraw-Hill Higher Education.

Gallahue, D., & Ozmun, J. (2002). *Understanding Motor Development: Infants, Children, Adolescents, Adults.* (5th ed.). McGraw-Hill.

Gentile, A. (1972). A working model of skill acquisition with application to teaching. *Quest*, *17*(1), 3–23.

Hardy, L. L., King, L., Farrell, L., Macniven, R., & Howlett, S. (2010). Fundamental movement skills among Australian preschool children. *Journal of science and medicine in sport / Sports Medicine Australia*, *13*(5), 503–8. doi:10.1016/j.jsams.2009.05.010.

Harper, R., Rodden, T., Rogers, Y., & Sellen, A. (2008). *Being human: human-computer interaction in the year 2020*. Retrieved September 30, 2014, from http://research.microsoft.com/en-us/um/cambridge/projects/hci2020/downloads/beinghuman_a4.pdf.

Haywood, K. M., & Getchell, N. (2009). *Life Span Motor Development* (5th ed.). Champaign.IL: Human Kinetics.

Hoffmann, K. B. (2009). *Domain-driven design in action designing an identity provider*. Retrieved August 31, 2014, from http://www.diku.dk/forskning/performance-engineering/Klaus/speciale.pdf.

Huber, J. (2012). *Applying educational psychology in coaching athletes* (1st ed.). Human Kinetics.

Jana, A. (2012). *Kinect for Windows SDK programming guide*. Packt Publishing. Retrieved October 30, 2013, from http://books.google.com/books?hl=en&lr=&id=7XqIvRDHVzkC&oi=fnd&pg=PT16&dq=Kinect+for+Windows+SDK+Programming+Guide&ots=EBYmNXUcvg&sig=LrUYvQOnK0yzxFJ4TMJVjBv0qzQ.

Johannesson, P. (2012). *A Design science primer* (pp. 33–36). CreateSpace Independent Publishing Platform.

Johnson, S. (2010). Daily reacharound: kinect reactions. Retrieved May 4, 2013, from http://www.g4tv.com/thefeed/blog/post/708449/daily-reacharound-kinect-reactions/.

Kalla, R. (2013). *Ps3-playstation-move-controller-angled-hand*. Retrieved May 4, 2013, from http://www.thebuzzmedia.com/gdc-sony-announces-move-eyetoy-based-motion-controller-for-ps3/ps3-playstation-move-controller-angled-hand/.

Khan, A., Moideen, F., Lopez, J., Khoo, W. L., & Zhu, Z. (2012). KinDectect : Kinect detecting objects. *Computers Helping People with Special Needs* : 588–595. doi:10.1007/978-3-642-31534-3_86.

Krog, D. S. (2010). *Movement programmes as a means to learning readiness*. University of South Africa. Retrieved April 25, 2013, from http://uir.unisa.ac.za/bitstream/handle/10500/3514/thesis_krog_s.pdf?sequence=1.

LeapFrog. (2013). *What is motor development?* Retrieved August 17, 2013, from http://www.leapfrog.com/en/leapfrog_parents/toddler/learning_for_life/what_is_motor_development.html.

Lee, J. C. (2008). Hacking the Nintendo Wii Remote. *IEEE Pervasive Computing*, *7*(3), 39–45. doi:10.1109/MPRV.2008.53.

Leeds Metropolitan university. (2004). *Fundamental movement development*. Retrieved April 24, 2013, from http://www.suffolklearning.co.uk/do_download.asp?did=950.

Linfo. (2007). *Command line interface definition.* Retrieved September 27, 2013, from http://www.linfo.org/command_line_interface.html.

Lubans, D., Morgan, P., Cliff, D., Barnett, L., & Okely, A. (2010). Fundamental movement skills in children and adolescents: review of associated health benefits. *Sports Med*, *40*, 1019–1035.

Mahalakshmi, T., Muthaiah, R., Swaminathan, P., & Nadu, T. (2012). Review Article : An Overview of Template Matching Technique in Image Processing. *Research Journal of Applied Sciences, Engineering and Technology*. Retrieved July 4, 2014, from http://maxwellsci.com/print/rjaset/v4-5469-5473.pdf.

Malcolm, G. (2011). *Kinect SDK: skeleton tracking*. Retrieved June 1, 2014, from http://cm-bloggers.blogspot.com/2011/06/kinect-sdk-skeleton-tracking.html.

Malina, R. M. (2003). Motor Development during Infancy and Early Childhood : Overview and Suggested Directions for Research. *International Journal of Sport and health science*. Retrieved April 22, 2013, from http://www.researchgate.net/publication/228978581_Motor_development_during_infancy_and_early_childhood_Overview_and_suggested_directions_for_research.

Matin, M., & Hands, B. (2003). Fundamental Movement Skills: Teachers' perspectives. *Australian Journal of EarlyChildhood*, *28*(4).

McCaughtry, N., Barnard, S., Martin, J., Shen, B., & Kulinna, P. H. (2006). Teachers' Perspectives on the Challenges of Teaching Physical Education in Urban Schools. *Research Quarterly for Exercise and Sport*, *77*(4), 486–497. doi:10.1080/02701367.2006.10599383.

Meyer, M., Helfert, M., Donnellan, B., & Kenneally, J. (2012). Applying Design Science Research for Enterprise Architecture Business Value Assessments. Design Science Research in Information Systems. Advances in Theory and Practice (pp. 108–121). Springer Berlin Heidelberg. doi:10.1007/978-3-642-29863-9_9.

Microsoft Corporation. (2013a). *Microsoft Xbox.* Retrieved May 4, 2013, from http://www.xbox.com/en-US/kinect.

Microsoft Corporation. (2013b). *Interface guidelines*. Retrieved April 20, 2013, from http://hackanui.com/resources/Human_Interface_Guidelines_v1.7.0.pdf.

Microsoft Developer Network. (2014). *Chapter 3: Architectural Patterns and Styles. MSDN*. Retrieved August 5, 2014, from http://msdn.microsoft.com/en-us/library/ee658117.aspx.

Millslagle, D. D. (2003). *Introduction to motor development*. Retrieved September 18, 2013, from http://www.d.umn.edu/~dmillsla/courses/motorlearning/documents/IntroductiontoMotorDevelopment.pdf.

Mitchell, B., McLennan, S., Latimer, K., Graham, D., Gilmore, J., & Rush, E. (2011). Improvement of fundamental movement skills through support and mentorship of class room teachers. *Obesity Research & Clinical Practice*, 2–6. doi:10.1016/j.orcp.2011.11.002.

Morris, G. D., & Steal, J. (1999). *Changing kid's games* (2nd ed., pp. 1–15). Human Kinetics Publishers.

National Ireland Curriculum. (2013). *Physical Development & Movement*. Retrieved May 5, 2013, from http://www.nicurriculum.org.uk/foundation_stage/areas_of_learning/physical_development/.

Nijholt, A., Pasch, M., Dijk, B. Van, Reidsma, D., & Heylen, D. (2011). *Whole Body Interaction*. (D. England, Ed.), 101–119. doi:10.1007/978-0-85729-433-3.

Nintendo Entertainment system. (2013). *Wii*. Retrieved May 4, 2013, from http://www.nintendo.com/wii.

NSW Department of Education and Training. (2000). *Get skilled : Get active* (pp. 16–17). Retrieved from http://www.healthykids.nsw.gov.au/downloads/file/teacherschildcare/Get_skilled_get_active_booklet.pdf.

Okely, A., Booth, M. L., & Patterson, J. (2001). Relationship of physical activity to fundamental movement skills among adolescents. *Med Sci Sports Exerc*, *33*, 1899–1904.

Olrich, T. W. (2013). Assessing Fundamental Motor Skills in the Elementary School Setting: Issues and Solutions. *Journal of Physical Education, Recreation & Dance*, (September), 37–41. doi:10.1080/07303084.2002.10607843.

OLX. (2013). *Play Station 3 com kit move*. Retrieved May 4, 2013, from http://cidaderiodejaneiro.olx.com.br/play-station-3-com-kit-move-iid-358786400.

OpenCV. (2014). *Neural networks*. Retrieved July 13, 2014, from http://docs.opencv.org/modules/ml/doc/neural_networks.html.

Otakismo. (2013). *Nintendo3DS*. Retrieved May 4, 2013, from http://otakismo.blogspot.com/2011/06/nintendo-3ds-e-os-riscos-da-inovacao.html.

Payne, V., & Isaacs, L. (2002). *Human Motor Development: A Life Span Approach.* (5th ed., pp. 434–435). Mountain view, California: Mayfield.

PDCE Admin. (2010). ECI – Module 1: *Typical Development*. Retrieved September 17, 2013, from http://blogs.ubc.ca/earlychildhoodintervention1/category/1-3-what-is-development-motor-development/.

Pica, R. (1997). Beyond physical development: Why young children need to move. *Young Children*, *52*(6), 4–11.

Portela, N. (2007). An assessment of the motor ability of learners in. Un*iversity o*f *Zululand*. Retrieved from http://uzspace.uzulu.ac.za/bitstream/handle/10530/110/An+Assessment+of+the+Motor+Ability+of+Leaners+-+N+Portela.pdf;jsessionid=382BB08D5C98BF5B2FCA73F110950924?sequence=1.

Rees, D. (2010). *Natural user interfaces as natural learning tools.* Retrieved September 30, 2014, from http://instructionaldesignfusions.wordpress.com/2010/11/15/natural-user-interfaces-as-natural-learning-tools/#comments.

Saffer, D. (2014). Chapter 1. Introducing Interactive Gestures. *O'Reilly Media*. Retrieved September 16, 2014, from http://www.safaribooksonline.com/library/view/designing-gestural-interfaces/9780596156756/ch01.html.

Schmidt, R. (2013). S*oftware Engineering*. (1st ed. : 11–12). Morgan Kaufmann.

Schmidt, R., & Lee, T. (2005). *Motor Control and Learning - 5th Edition: A Behavioral Emphasis* (5th ed.). Human Kinetics.

Sezgin, M. (2012). Intelligent User Interfaces. *Signal Processing and Communications Applications Conference (SIU), 2012 20th* (p. 1). IEEE. doi:10.1109/SIU.2012.6204419.

SkyNETitsolutions. (2011). *Natural User Interfaces: Voice Touch and Beyond*. Retrieved April 28, 2013, from http://www.skynetitsolutions.com/blog/natural-user-interfaces.

Sommerville, I. (2001). S*oftware Engineering* (9th Ed.). Addison-Wesley.

Sony Computer Entertainment. (2013). *Playstation 3*. Retrieved May 4, 2013, from http://za.playstation.com/ps3/.

Sport New Zealand. (2012a). *Locomotor skill dodging*. Retrieved September 12, 2013, from http://www.sportnz.org.nz/Documents/Young People/E_5620-3_SPC_A4_2_locomotor-ff_WEB_dodging.pdf.

Sport New Zealand. (2012b). *Striking with the feet*. Retrieved September 12, 2013, from http://www.sportnz.org.nz/Documents/Young People/Q_5620-3_SPC_A4_4_manipulative-ff_WEB_feet.pdf.

Sport New Zealand. (2012c). *Striking with an implement* (pp. 292–328). Retrieved from http://www.sportnz.org.nz/Documents/Young People/R_5620-3_SPC_A4_4_manipulative-ff_WEB_implement.pdf.

Sport New Zealand. (2012d). *Fundamental movement skills among children in New Zealand* (pp. 1–8). Wellington: Retrieved from www.sparc.org.nz.

Stergiou, C., & Siganos, D. (1996). N*eural networks*. Retrieved July 14, 2014, from www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#What is a Neural Network.

Tanaka, K., Parker, J., Baradoy, G., Sheehan, D., Holash, J. R., & Katz, L. (2012). A Comparison of Exergaming Interfaces for Use in Rehabilitation Programs and Research. *Loading...*, *6*(9), 69–81.

Taylor, R. N., Medvidovic, N., & Dashofy, E. M. (2008). *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, Inc.

Theeboom, M., De Knop, P., & Weiss, M. (1995). Motivational climate, psychological responses and motor skill development in children's sport: a field-based intervention study. *J Sport Exerc Psychol*, *17,* 294–311.

Tsichritzis, D. (1998). *The Dynamics of Innovation,"in Beyond Calculation: The Next Fifty Years of Computing*. (Denning, P. J. &. Metcalfe, R. M., Eds.: 259–265). Copernicus Books.

Tullis, T., & Albert, B. (2008). *Measuring the user experience* (pp. 65–70). Morgan Kaufmann Publishers.

Vaishnavi, V., & Kuechler, W. (2004a). *Design Science Research in Information Systems*. Retrieved April 14, 2013, from http://www.desrist.org/design-research-in-information-systems.

Vaishnavi, V., & Kuechler, W. (2004b). *Design Science Research in Information Systems*. Retrieved May 8, 2013, from Kuechler, W.

Vaughan-Nichols, S. J., & Douglas, S. (2009). Game-Console Makers Battle over Motion- Sensitive Controllers. *Computer*, *42*(8), 13–15. doi:10.1109/MC.2009.260.

Webb, J., & Ashley, J. (2012). *Beginning Kinect Programming with the Microsoft Kinect SDK* : 167–176. Apress.

Wigdor, D., & Wixon, D. (2011). *Brave NUI world*. (R. Roumeliotis & D. Bevans, Eds.) (pp. 3–15). Morgan Kaufmann.

Williams, C. L., Carter, B. J., Kibbe, D. L., & Dennison, D. (2009). Increasing physical activity in preschool: a pilot study to evaluate animal trackers. *Journal of nutrition education and behavior*, *41*(1), 47–52. doi:10.1016/j.jneb.2008.03.004.

Yampolskiy, R., & Govindaraju, V. (2007). Direct and Indirect Human Computer Interaction Based Biometrics. *Journal of Computers*, *2*(10), 76-88.

Zeng, W., & Zhang, Z. (2012). Multimedia at Work Microsoft Kinect Sensor and Its Effect. *MultiMedia, IEEE*, *9*(2), 4–10. doi:10.1109/MMUL.2012.24.

Appendix A: Experiment 1

Task 1

Perform fifteen correct attempts at the sub-skill of arms moving back.

Perform fifteen incorrect attempts at the sub-skill of arms moving back.

Task 2

Perform fifteen correct attempts at the sub-skill of knees bent.

Perform fifteen incorrect attempts at the sub-skill of knees bent.

Task 3

Perform fifteen correct attempts at the sub-skill of arms stretched.

Perform fifteen incorrect attempts at the sub-skill of arms stretched.

Task 4

Perform fifteen correct attempts at the sub-skill of legs straight.

Perform fifteen incorrect attempts at the sub-skill of legs straight.

Task 5

Perform fifteen correct attempts at the sub-skill of controlled landing.

Perform fifteen incorrect attempts at the sub-skill of controlled landing.

Appendix B: Experiment 2

Task 1

Perform fifteen correct attempts at a valid jump (all sub-skills should be attempted correctly at the same time)

Task 2

Perform fifteen attempts at an invalid jump (combination of correct and incorrect attempts at sub-skills at the same time) in the following manner:

- Perform four attempts at an invalid jump with arms not straight in the air and legs not being straight.
- Perform five attempts at an invalid jump with the landing not controlled.
- Perform one attempt at an invalid jump with the arms not straight in the air and the land not controlled.
- Perform three attempts at an invalid jump with the arms not pulled back, the arms not straight in the air and the land not controlled.
- Perform two attempts at an invalid jump with the arms not straight in the air and the arms not pulled back.