Distributed Technology-Sustained Pervasive Applications –
Kim J.L. Nevelsteen
Distributed Technology-Sustained Pervasive Applications

Kim J.L. Nevelsteen
Technology-sustained pervasive games, contrary to technology-supported pervasive games, can be understood as computer games interfacing with the physical world. Pervasive games are known to make use of ‘non-standard input devices’ and with the rise of the Internet of Things (IoT), pervasive applications can be expected to move beyond games. This dissertation is requirements- and development-focused Design Science research for distributed technology-sustained pervasive applications, incorporating knowledge from the domains of Distributed Computing, Mixed Reality, Context-Aware Computing, Geographical Information Systems and IoT. Computer video games have existed for decades, with a reusable game engine to drive them. If pervasive games can be understood as computer games interfacing with the physical world, can computer game engines be used to stage pervasive games? Considering the use of non-standard input devices in pervasive games and the rise of IoT, how will this affect the architectures supporting the broader set of pervasive applications? The use of a game engine can be found in some existing pervasive game projects, but general research into how the domain of Pervasive Games overlaps with that of video games is lacking. When an engine is used, a discussion of, what type of engine is most suitable and what properties are being fulfilled by the engine, is often not part of the discourse. This dissertation uses multiple iterations of the method framework for Design Science for the design and development of three software system architectures. In the face of IoT, the problem of extending pervasive games into a fourth software architecture, accommodating a broader set of pervasive applications, is explicated. The requirements, for technology-sustained pervasive games, are verified through the design, development and demonstration of the three software system architectures. The scaling up of the architecture to support distributed pervasive applications, is based on research in the domain of Virtual Worlds and IoT. The results of this dissertation are: the aligning of the Pervasive Games research domain with that of Virtual Worlds, the mapping of virtual time and space to physical counterparts, the scaling up of pervasive games to distributed systems, and the explication of the problem of incorporating IoT into pervasive applications. The implication of this dissertation is to ensure that pervasive games are not left reinventing existing technologies.
Sammanfattning

Teknikförmedlade verklighetsspel (technology-sustained pervasive games), i motsats till teknikstödda verklighetsspel (technology-supported pervasive games), kan förstås som dataspelets gränssnitt mot den fysiska världen. Verklighetsspel games är kända för att använda sig av ‘icke-standardiserade inmatningsenheter’ och med ökningen av Sakernas Internet (Internet of Things) (IoT), kan verklighetsapplikationer (pervasive applications) förväntas gå längre än verklighetsspel. Denna avhandling omfattar krav- och utvecklingfokuserad (Design Science) forskning för distribuerad teknik omfattande verklighetsspel, som innehåller kunskap från områdena distribuerad databehandling (Distributed Computing), blandad realitet (Mixed Reality), kontextmedveten databehandling, geografiska informationssystem och IoT. Dataspel har funnits i decennier, ofta med en återanvändbar spelmotor för att driva dem. Om verklighetsspel kan förstås som dataspel med gränssnitt mot den fysiska världen, kan då dataspelsmotorer användas för att iscensätta verklighetsspel? Med tanke på användningen av icke-standardiserade inmatningsenheter i verklighetsspel och den tilltagande mängden IoT tillämpningar, hur kommer detta att påverka arkitekturen som stöder verklighetsspel?

Dedicated to . . .
the Wehlou family, my extended family,
for without their influence and friendship my life would be unrecognizable.

And . . .
to Emma, for loving me through the most difficult time of my life
(to which earning a PhD certainly contributed),
and for giving me my baby boy, Viggo.
Acknowledgements

First and foremost, I wish to express my gratitude to my supervisors, Professor Theo Kanter and Dr. Rahim Rahmani, who provided me the time and space to realize my PhD; their patience and belief in my ability made it possible to produce the bulk of the publications herein and the dissertation itself. My sincerest gratitude goes to Annika Waern, my supervisor for the first two years of the PhD; it was a truly interesting learning experience, with an introduction to pervasive games, research, and more (e.g., the carpooling is memorable). A special thanks to Fredrik Charpentier Ljungqvist for his guidance on the rules and regulations of being a PhD student. A big thank you goes to Paul Johannesson for his constructive pre-Doc criticism, and the writing of *An Introduction to Design Science*, making it easier for me to clarify my methodology. I want to express my appreciation for the existence of Google Scholar, ACM Digital Library and other digital resources of today; I can not imagine doing cross-domain research without such tools (actually I can e.g., try finding not-so-recent GIS publications online).

This dissertation was made possible by the Department of Computer and Systems Sciences (DSV) and a grant from the Swedish Governmental Agency for Innovation Systems to the Mobile Life Vinn Excellence Center. I wish to thank DSV for backing me through some tough times: Åsa Smedberg for being the mediator, Louise Barkhuus for creating a good working environment at DSV, Harko Verhagen for being Harko, Nam Aghaee for energetic supportive student advice, and the administration team at DSV for handling the paper trail. From my immediate research group, Immersive Networking, I especially want to thank Xiao Bin, for his knowledge and drive about publishing, his kindness (yummy FOOD, but no pancake), cheerfulness and support; Yuhong Li for her kindness and Hasibur Rahman for his jolly spirit. Thank you to Mobile Life for the in-depth introduction into Human-Computer Interaction and amazing excursions the first two years. Thank you Barry Brown for positive negotiations in tough times, Maria Holm for her kindness and business sense, Oskar Juhlin for being himself and Kristina Höök for running the center. A special thanks to Syed Naseh, Zhang Yanqing (Celia), Mattias Jacobsson and Mudassir Ahmad Mughal for their unwavering kindness.

In addition, I wish to also extend my thanks to . . .

---

1 All literature and notes for this dissertation were collected digitally; no publications were printed for reviewing or note taking; books were all obtained digitally, except for those only available in print and not in an accessible library i.e., a minimum of trees have been hurt during the production of this dissertation. I apologize for the biological footprint of the computer systems that were (ab)used during the production.
... all the people at the Swedish Institute of Computer Science (SICS), for the educational and inspiring lunch talks. A special thanks to Magnus Boman and Baki Cakici for interesting philosophical talks and advice in troubled times. Thank you Joakim Eriksson for the input and resources for building a game engine based IoT prototype. A special thanks to Jakob Axelsson, Emmanuel Frécon, Markus Bylund, Pär Hansson, Bengt Ahlgren, Niklas Ekström, Šarūnas Girdzijauskas, Simon Duquennoy, and Martin Nilsson, for letting me pick their brains. Thank you to the SICS CRIT team: Thomas Ringström, Orc Lönn and Alex Bustamante, for their kindness and helping me geek out on technology.

With miscellaneous affiliations: A big special thanks to Jim Wilenius for being my sounding board over the last five long years. Thank you: Joel Norman for his mathematical knowledge and tolerating my endless rants; Paolo Santin for his continuous support and statistical knowledge; ‘Graphics’ Martin Ericsson for being the silent one, but still answering my emails every once in a while; Leif Oppermann for his critical review of BOOK I; Newton Lee for his amazing kindness and expertise on publishing venues; Inger Ekman for her reviews, energy and enthusiasm; Frans Mäyrä for his professionalism, wisdom and encouragement; and, Justin Pearson for actually recommending me to do a PhD with Annika.

I would like to recognize the following people for their kindness at some point in the last five years\(^1\). Interactive Institute: Susanne Mattsson, Anders Wahlfridsson, Annelise de Jong, Britt-Mari (Brittis) Andersson and Christina von Dorrien. Mobile Life: Elena Schulte, Karl-Petter Åkesson, Marie Denward, Elena Márquez Segura, Sergio Gayoso Fernández, Elin Önnevall, Zeynep Ahmet, Gail Pailin Johnsson, Nicholas Belloni, Morvarid Kashanipour, Johanna Mercurio, Marcus Lundén, Katja Grufberg, Pedro Sanches, Jordi Solsona, Jon Back, Victor Guerrero Corbi, Jinyi Wang, Tom Homewood, Sara Ljungblad, Goranka Zoric, Elsa Vaara, Anna Ståhl, Sebastian Büttner, Claus Weymann, Pedro Ferreira and Sophie Landwehr Sydow. SICS: Gautier Berthou, Joel Höglund, Karin Fohlstedt and Kersti Hedman. DSV/ACT: Jozef Swiatycki (especially for his great sense of humor), Angela Westin, Olle Karlsson, David Söderström, Aware Mustafa Tahir, Staffan Åhlvik, Magnus Johansson, Henrik Warpefelt, Björn Strååt, Isak Karlsson and Zhao Jing. And, Yadmaa and Baaska from Pärla Sushi, for their support and continuous supply of Bibimbab.

Last, but certainly not least, I would like to thank my family and friends who indirectly contributed to this work through moral support; a list of everyone would be too long enumerate here.

---

\(^1\)With such lists, I’m bound to have forgotten someone, my apologizes, mea culpa.
List of Publications

The following papers, referred to in the text by their Roman numerals, are included in this thesis.

PAPER I :  **GDD as a Communication Medium**  
ISBN: 978-951-44-8705-7

PAPER II : **Athletes and Street Acrobats:**  
*Designing for Play as a Community Value in Parkour*  
DOI: 10.1145/2207676.2208528

BOOK I :  **A Survey of Characteristic Engine Features for Technology-Sustained Pervasive Games**  
DOI: 10.1007/978-3-319-17632-1

PAPER V : **Virtual World, Defined from a Technological Perspective, and Applied to Video Games, Mixed Reality and the Metaverse**  
Kim J. L. Nevelsteen [submitted 2015].

PAPER V bis : **Virtual World, a Definition Incorporating Distributed Computing and Instances**  
DOI: 10.1007/978-3-319-08234-9_44-1
PAPER VI: Spatiotemporal Modeling of a Pervasive Game
Kim J. L. Nevelsteen [submitted 2016].

PAPER VII: Comparing Properties of Massively Multiplayer Online Worlds and the Internet of Things
Kim Nevelsteen, Theo Kanter, Rahim Rahmani [submitted 2016].

Not Included in this Dissertation

PAPER: Applying GIS Concepts to a Pervasive Game: Spatiotemporal Modeling and Analysis Using the Triad Representational Framework
E-ISSN: 1694-0814

Reprints were made with permission from the publishers.
Author’s Contributions

The contribution of each published work is summarized in the following list:

PAPER I

*GDD as a Communication Medium*

Requirements research and the original architectural design of the GDD was done by this author. PAPER I is done in collaboration with Sergio Gayoso Fernández; a master student under supervision at the time. With the help of Annika Waern and this author, Sergio Gayoso adapted the architectural design, in three iterations, including the staged case studies and phenomenological interviews. The GDD proved to model a persistent communication technology, of which the requirements analysis, design and the notion of a ‘view’ are a contribution. Collaborative user editing via the Internet was not mainstream at the time and the ‘view’ concept is still not wide spread today.

PAPER II

*Athletes and Street Acrobats: Designing for Play as a Community Value in Parkour*

The design of a pervasive service to satisfy project stakeholders and the Parkour community included: the design of an architecture to sustain a pervasive service, called Traveur, and a meet-up map function. The design of the service itself was not done by this author, but by others in the research group. The design of the architecture was done by Annika Waern (our research group) and Joel Westerberg (Street Media 7), with added input from this author. Design of the map functionality was done solely by this author, in collaboration with the author’s supervisor at the time, Annika Waern. Final redesign and trials of the service were done by Elena Balan, under the supervision of Annika Waern and with the help of this author. Traveur, of PAPER II, highlights that “heterogeneity is the norm” i.e., that having both heterogeneous servers and clients is becoming far more commonplace. Traveur served as exploratory research into pervasive and context-aware computing, of which the requirements analysis and design are a contribution; poor architectural design lead to broadening the search for subsequent technological solutions to include the domain of Computer Games.
BOOK I

A Survey of
Characteristic Engine Features for Technology-Sustained Pervasive Games

This author is the sole author of the work. Chapter 2 of BOOK I is an extensive systematic review into pervasive games. The resulting feature set is a contribution describing characteristic features of a would-be pervasive games engine. These features can be considered a set of informal requirements from which a set of formal requirements can be drawn. Using the feature set, a virtual world engine was chosen as being in the same ‘product line’ [Bass, Clements, and Kazman 2013] as a would-be pervasive games engine, based on the shared trait of a persistence. In Chapter 3 of BOOK I, the component feature set and the choice of a virtual engine as pervasive engine are verified through the case study of a pervasive game called CN:H. BOOK I highlighted that no usable definition for a ‘virtual world’ existed and that an integrated model, mapping time and space from the physical to the virtual, and vice versa, was missing.

PAPER V

Virtual World, Defined from a Technological Perspective, and Applied to Video Games, Mixed Reality and the Metaverse

This author is the sole author of the work. Explicit from the title, the contribution of PAPER V is that it provides a definition for a ‘virtual world’ from a technological perspective, rather than a cultural one; all underlying properties of the definition are defined in detail. Less explicit contributions are: the determining of what constitutes a virtual world when dealing with distributed and possibly disconnected architectures; how a virtual world relates to virtual and mixed reality; and, an ontology showing the relationship between complimentary terms and acronyms. To verify the definition, it is used to classify contemporary technologies e.g., the video game called Destiny.

PAPER V bis

Virtual World, a Definition Incorporating Distributed Computing and Instances

This author is the sole author of the work. PAPER V bis is an excerpt of PAPER V above. It only includes the definition for a ‘virtual world’ and its relationship to mixed reality. The purpose of including this publication in the dissertation is so that the definition would be peer reviewed by the time of the dissertation defense, while the full length article of PAPER V undergoes review and rebuttal over the next several months.
PAPER VI

Spatiotemporal Modeling of a Pervasive Game

This author is the sole author of the work. In the domain of Pervasive Games, research exists tying the virtual to the physical, but an integrated spatiotemporal conceptual model, for mapping the virtual to the physical and vice versa, is lacking. Because Geographical Information Systems and Pervasive Games both make use of the Earth’s geography, their problem domains overlap i.e., research found in Geographical Information Systems, on how to model the physical world, can be exapted to Pervasive Games. The contribution of PAPER VI is that it provides a integrated conceptual model through the exaptation of Peuquet’s Triad Representational Framework, combining it with Dix et al.’s three types of space and a notion of time.

PAPER VII

Comparing Properties of Massively Multiplayer Online Worlds and the Internet of Things

Research and drafting of property requirements relating Massive Multiplayer Online Worlds and the Internet of Things is done by this author. Conceptual work on decentralizing a game engine for use with Internet of Things is done by this author, in collaboration with supervisors, Theo Kanter and Rahim Rahmani. Each of the case studies is provided by a single author: MediaSense by Theo Kanter, Immersive Networking by Rahim Rahmani and Virtual Worlds as a “behind the scenes” resource by this author.

PAPER VII surveys the domain of MMOWs (large scale virtual worlds) for properties that are affected by scaling an architecture, and then evaluates how these are dealt with in the domain of IoT. The contribution of PAPER VII is the problem explication of scaling architectures for MMOWs and IoT; six properties related to scaling are discussed. That device and system heterogeneity is an issue in the domains of Virtual Worlds and IoT, is particularly relative to this dissertation.
# Contents

Abstract v  
List of Publications xiii  
List of Figures xxi  
List of Tables xxi  

## Part I Thesis  

### 1 Introduction  
1.1 Problem Statement ................................. 1  
1.2 Research Question ................................. 2  
1.3 Research Approach ................................. 2  
1.4 Contributions ...................................... 4  
1.5 Dissertation Structure ............................ 6  

### 2 Research Strategies / Methods  
2.1 This Dissertation is Design Science Research ........ 10  
2.2 Multiple Iterations of Method Framework ............... 11  
2.2.1 Design and Development of a New Medium for a GDD 11  
2.2.2 Traveur: Pervasive Architecture and Map Functionality 11  
2.2.2.1 Four Distinguishable Phases .................. 12  
2.2.2.2 Map Functionality ............................ 14  
2.2.3 Pervasive Games Architecture: Pervasive MOO ...... 16  
2.2.3.1 Survey of Pervasive Games and Technologies 16  
2.2.3.2 Virtual World Engine Staging a Pervasive Game 16  
2.2.3.3 Integrated Spatiotemporal Conceptual Model 17  
2.2.3.4 A Definition for ‘Virtual World’ ................ 17  
2.2.4 Aligning Research of MMOWs with that of IoT ....... 18  

### 3 Related Work  

4 Design of Three Software System Architectures 23
  4.1 A Persistent Communication Technology ......................... 24
  4.2 Design of a Pervasive Parkour Game; shifting requirements and mashed-up components .......... 24
    4.2.1 Architectural Back-end ................................ 25
    4.2.2 Architectural Front-end ................................ 27
    4.2.3 Heterogeneity is the Norm ............................... 28
  4.3 Pervasive Game Architecture .................................... 29

5 Mixed Reality and Scalability 33
  5.1 ‘Virtual World’ Defined and Applied ............................. 33
    5.1.1 Is the GDD a Virtual World? ............................... 34
    5.1.2 Is Traveur a Virtual World? ............................... 34
    5.1.3 Pervasive Games Overlap with Virtual Worlds? .......... 35
    5.1.4 One World or a World of Worlds? ......................... 37
  5.2 Mapping Between the Virtual and Physical ....................... 37
  5.3 Scaling up the Architectures to Distributed Systems .......... 40
    5.3.1 Client / Server ........................................... 40
    5.3.2 A Centralized Server Cluster ............................. 42
    5.3.3 Heterogeneous Servers .................................... 42
    5.3.4 Heterogeneous Clients .................................... 44
    5.3.5 Peer-to-Peer .............................................. 45
  5.4 Distributed Pervasive Applications, Incorporating IoT .......... 46
    5.4.1 Resource Utilization ..................................... 46
    5.4.2 Availability .............................................. 47
    5.4.3 Responsiveness ............................................ 47

6 Evaluation 49
  6.1 Validating the Research Question ................................ 49
  6.2 Verifying the Research Methodology .............................. 49
  6.3 Evaluating the Design of the Architectures ...................... 50
    6.3.1 The GDD as a Basis for Pervasive Applications .......... 50
    6.3.2 The Heterogeneity of a Pervasive Service ............... 51
    6.3.3 Evaluating the Feature Set for Pervasive Games ........ 52
  6.4 Evaluating the Combining of Results ............................. 52
    6.4.1 Overlap of Pervasive Games and Virtual Worlds ......... 52
    6.4.2 Evaluating the Virtual / Physical Mapping ............... 55
    6.4.3 Scaling Up Architectures and Incorporating IoT .......... 57

7 Conclusion 59

8 Future Work 63
List of Figures

1.1 Overlapping domains of Distributed Pervasive Applications. . 5
1.2 Dissertation ‘road map’ . . . . . . . . . . . . . . . . . . . . 6

2.1 Method Framework for Design Science Research . . . . . . . 10
2.2 ‘Volt’ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.3 Two images of the map functionality napkin design . . . . . . 15

4.1 Drawing of the Traveur architecture . . . . . . . . . . . . . 26
4.2 Sketch of the Traveur architecture . . . . . . . . . . . . . . 26
4.3 Traveur’s change of the dependency graph . . . . . . . . . . 28
4.4 Coextensive worlds . . . . . . . . . . . . . . . . . . . . . . 32

5.1 GDD architecture . . . . . . . . . . . . . . . . . . . . . . . 41
5.2 CN:H architecture . . . . . . . . . . . . . . . . . . . . . . . 41
5.3 Traveur architecture, after change of dependency graph . . . 43
5.4 Hybrid architecture . . . . . . . . . . . . . . . . . . . . . . 45

6.1 Integrated spatiotemporal conceptual model applied to CN:H . 55

List of Tables

5.1 Classification of the architectures according to PAPER V . . . 35
Part I

Thesis
The Five Orders of Ignorance [Armour 2000, original bold]

“Since we are computer folk, we start counting from zero, rather than one. . . .

0th Order Ignorance (0OI)– Lack of Ignorance.
I have 0OI when I know something and can demonstrate my lack of ignorance in some tangible form, . . .

1st Order Ignorance (1OI)– Lack of Knowledge.
I have 1OI when I don’t know something and can readily identify that fact. 1OI is basic ignorance. . . .

2nd Order Ignorance (2OI)– Lack of Awareness.
I have 2OI when I don’t know that I don’t know something. That is to say, not only am I ignorant of something (for instance I have 1OI), I am unaware of this fact. I don’t know enough to know that I don’t know enough. . . .

3rd Order Ignorance (3OI)– Lack of Process.
I have 3OI when I don’t know a suitably efficient way to find out I don’t know that I don’t know something. . . . If I have 3OI, I don’t know of a way to find out there are things I don’t know that I don’t know. Therefore, I can’t change those things I don’t know that I don’t know into either things that I know, or at least things I know that I don’t know, as a step toward converting the things I know that I don’t know into things I know. . . .

4th Order Ignorance (4OI)– Meta Ignorance.
I have 4OI when I don’t know about the Five Orders of Ignorance. I no longer have this kind of ignorance, and now, neither, dear reader, do you.”
1. Introduction

Pervasive games [Nevelsteen 2016a] are becoming more common place e.g., with Google, the multinational company, staging a pervasive game called Ingress [Niantic Labs 2013], and applications (apps) like Zombies, Run! [Six to Start 2012] becoming more mainstream. According to Benford, Magerkurth, and Ljungstrand [2005], “pervasive games extend the gaming experience out into the real [physical] world”. Whereas Montola [2005] states that a pervasive game “is a game that has one or more salient features that expand the contractual magic circle of play socially, spatially or temporally” i.e., “expand the boundaries of play” [Oppermann 2009]. In her definitive work, Nieuwdorp [2007, original italics] derives that pervasive games can be discussed from two perspectives, a technological one, “that focuses on computing technology as a tool to enable the game to come into being” or a cultural one, “that focuses on the game itself”. Pervasive games can be classified into those which are ‘technology-sustained’, relying on computer simulation to maintain game state and react to player activities; these games can be understood as “computer games interfacing with the physical world” [Montola, Stenros, and Waern 2009, p.164]. Contrary to technology-sustained pervasive games, ‘technology-supported’ games are those where not all game activities are supported by information technology [Montola, Stenros, and Waern 2009] i.e., do not necessarily require a game engine. Pervasive games are known to make use of ‘non-standard input devices’ [Nieuwdorp 2007] and with the rise of the Internet of Things (IoT) [Gartner 2014], pervasive applications can be expected to move beyond games. This dissertation is requirements- and development-focused Design Science research [Johannesson and Perjons 2014, p.79] for distributed technology-sustained pervasive applications, incorporating knowledge from the domains of Distributed Computing, Mixed Reality, Context-Aware Computing, Geographical Information Systems (GIS) and IoT.

1.1 Problem Statement

Computer video games have existed for decades, with reusable game engines to drive them; the major incentive for employing a reusable game engine being reduced development time and cost [Lewis and Jacobson 2002; Bass,
Clements, and Kazman 2013. Currently, there are no reusable game engines available for pervasive games; without such engines, developers are left continually reinventing existing technologies. If technology-sustained pervasive games can be understood as computer games interfacing with the physical world, can computer game engines be used to stage a pervasive game? And, can advances from the domain of Computer Video Games to be used to scale up pervasive games to distributed systems? To answer these questions, a requirements analysis for pervasive games must be performed, an architecture found that correlates to those requirements, and an analysis done to verify if existing architectural properties actually overlap with the requirements. According to Jonsson et al. [2007], pervasive games need a sensory system to monitor the physical world; access to the IoT could potentially serve as such a sensory system. Considering the use of non-standard input devices in pervasive games and the rise the IoT, how will this affect pervasive games architectures, allowing for a broader set of pervasive applications?

1.2 Research Question

Given the domain and the problem statement, the research question for this dissertation is then:

To allow advances from the domain of Computer Video Games to be used to scale up pervasive games to distributed systems, can engine technology from the domain of Computer Video Games be repurposed to stage technology-sustained pervasive games? Considering the use of non-standard input devices in pervasive games and the rise of Internet of Things, how will this affect the architectures supporting the broader set of pervasive applications?

These research questions are answered by first doing a requirement analysis into pervasive games, and then expanding the requirements to pervasive applications. Advances in the other domains, mentioned above, are taken into account during the analysis and design. Three software system architectures are created in four iterations to verify the requirements and provide input for further research.

1.3 Research Approach

This dissertation incorporates knowledge from the domains of Distributed Computing, Mixed Reality, Context-Aware Computing, GIS and IoT. Because the solution maturity in the other domains (e.g., Computer Video Games and GIS)
is high and the application domain maturity of Pervasive Games is low, this dissertation makes use of several exaptations; an ‘exaptation’ being the extension of a known solution to a new problem [Johannesson and Perjons 2014, p.10]. Four iterations of the method framework for Design Science [Johannesson and Perjons 2014] are used for the design and development of three software system architectures: the Game Design Document (GDD), a new game design and communication medium; Traveur, a pervasive game turned pervasive service due to fluid requirements; and, Codename: Heroes (CN:H), a “long-term pervasive game”. In the face of IoT, the fourth iteration, explicates the problem, of extending pervasive games into a broader set of pervasive applications.

I: The first architecture was a design exercise on how to model a GDD medium, in the form of a ‘living document’ (see Section 4.1). In hindsight, the project proved to be an exploratory case study; the architecture, a persistent communication technology (see Section 5.1.1), and the ‘view’ concept, an influence for subsequent iterations. Results of this iteration can be found in PAPER I.

II: In this iteration, an architecture for Traveur was designed, a pervasive service (originally a pervasive game) and a map function for that service. An article pertaining to the architecture was created in draft, but never published; to compensate, it is discussed in depth below (see Section 4.2). Redesign of the pervasive service itself and the map function was published in PAPER II. This project exemplifies shifting design requirements, heterogeneity as the norm, and context-awareness.

III: The most extensive architecture herein, for CN:H, was with the aim of designing an architectural back-end for a technology-sustained pervasive game (see Section 4.3). Chapter 2 of BOOK I is a requirements analysis for pervasive games through literature review; Chapter 3 is a demonstration of the requirements and resulting architecture as case study; and Chapter 4 is a summary of the challenges and open issues for pervasive games, which influenced the final iteration IV.

IV: A virtual world that supports a massive number of players, referred to as a Massively Multiplayer Online World (MMOW), is often enabled by distributed architectures. PAPER VII explicates the problem of scaling architectures for a MMOW and IoT. Because the domain of Pervasive Games is aligned with the domain of Virtual worlds herein, results from PAPER VII are combined with the open issues from BOOK I, so as to explicate the problem of extending pervasive games into a broader set of pervasive applications incorporating IoT. This then leads into the future work of Chapter 8.
Having completed CN:H, in Iteration III, it was clear that a model mapping the virtual to the physical (and vice versa) was lacking, as well as a definition for ‘virtual world’, notwithstanding that so-called ‘virtual world engines’ exist. A definition for a ‘virtual world’ is obtained using grounded theory in PAPER V, and because, research in pervasive games is aligned with that of virtual worlds in BOOK I (based on the persistence trait), concepts from PAPER V (e.g., mixed reality and distributed computing) can be subsequently applied to pervasive games. Using concepts borrowed from the domain of GIS, PAPER VI provides an integrated spatiotemporal conceptual model that maps the virtual to the physical and vice versa.

1.4 Contributions

In its entirety this dissertation provides requirements for technology-sustained pervasive games and an explicated problem, from which requirements can be formed, for distributed pervasive applications; requirements are verified by designs and feasibility implementations. In addition to the contributions found in each of the individual publications (see the Author’s Contributions p. xv), the dissertation contributions include the following points:

- Containing the design and development of three software system architectures, showing how each influenced the requirements for technology-sustained pervasive games i.e., the requirements analysis and design of the GDD and Traveur influencing CN:H, including game master interfaces, heterogeneous servers, context-awareness and mixed reality.

- Using PAPER V, it is shown that domain of Pervasive Games overlaps with that of Virtual Worlds, except for a discrepancy based on mixed reality; the majority of properties defining a virtual world are characteristics that are preferable to a would-be pervasive games engine. This result is a substantial improvement over choosing a virtual world engine based solely on the overlapping trait of persistence, as in BOOK I.

- By aligning pervasive games with that of virtual worlds, advances in architecture, from the domain of Virtual Worlds, are used to scale up pervasive games architectures to distributed systems.

- Using results from PAPER VII, pervasive games are extended to pervasive applications, incorporating IoT; properties pertaining to scalability from the domain of MMOWs and IoT are compared and the result used to explicate the problem of scaling architectures for pervasive applications.
The domain of Distributed Pervasive Applications is linked to the domains of Distributed Computing, Mixed Reality, Context-Aware Computing, GIS and IoT (see Figure 1.1). PAPER I, BOOK I, PAPER V and PAPER VII all highlight challenges surrounding Distributed Computing. PAPER V ties Virtual Worlds to Mixed Reality, and BOOK I, with the help of PAPER V, shows Pervasive Games to overlap with both the domain of Virtual Worlds and Mixed Reality. Traveur, the pervasive service in PAPER II, makes heavy use of Context-Aware Computing, and BOOK I links this to Pervasive Games. PAPER VI ties the domain of GIS to Pervasive Games based on Earth’s geography. And, PAPER VII expands the concept of pervasive games to pervasive applications, tying all the above to the domain of IoT.

Figure 1.1: Overlapping domains of Distributed Pervasive Applications.
1.5 Dissertation Structure

This dissertation is structured (see Figure 1.2) such that: the introduction, you have just read. Research strategies and methods are discussed in Chapter 2, with one underlying section for each iteration of the method framework. Related work is presented in Chapter 3. Chapter 4 contains three software system architectures in three iterations (Iteration I, II & III), one section for each iteration. In Chapter 5 the mixed reality of pervasive games is dealt with and architectures are scaled up to distributed systems. The final section of that chapter (Section 5.4) is Iteration IV, incorporating IoT. Chapter 6 validates the research question, verifies the research methodology and evaluates the work in Chapter 4 and 5. Chapter 7 is the conclusion, that summarizes what has been achieved in this dissertation. And, Chapter 8 is the future work section, which discusses the most direct extensions of this dissertation. Some of the future work follows directly on PAPER VII and Section 5.4, while some extend from BOOK I.

![Figure 1.2: Dissertation ‘road map’ - Visual depiction relating publications; organized in columns according to which iteration the publication or project was in, with arrows showing the flow of influence between projects.](image-url)
Chapter 4 contains three software system architectures in three iterations, one section for each iteration. The first architecture discussed (see Section 4.1 for Iteration I) is that of the GDD in PAPER I. The GDD proved to model a persistent communication technology, of which the requirements analysis, design and ‘view’ concept influenced the Traveur project. Properties of a persistent communication technology influenced the definition of a ‘virtual world’ in PAPER V (marked with the working title WorldDEF in Figure 1.2). Because the architecture of Traveur was never published, a detailed account of the Traveur architecture, and its connection to PAPER II, is presented in Section 4.2 (Iteration II). The requirements analysis and design of Traveur influenced CN:H of BOOK I; poor architectural design lead to broadening the search for subsequent technological solutions to include the domain of Computer Games. Other concepts from Traveur that influenced subsequent iterations were heterogeneous servers, context-awareness and mixed reality. The largest iteration (Iteration III) contains the architecture for CN:H. BOOK I contains the requirements analysis for a pervasive games architecture, and a case study evaluating the architecture (see Section 4.3). Pervasive games are shown to overlap with virtual worlds, via a persistence trait. And, that no usable definition for a ‘virtual world’ existed was highlighted.

PAPER V provides a definition, and in Chapter 5 it is used to show in detail the overlap between pervasive games and virtual worlds (see Section 5.1). Because pervasive games are not required to have Virtual Spatiality, clarification is needed on how to deal with their mixed reality. Using the results from PAPER VI (marked with the working title TimeSpace in Figure 1.2), time and space are mapped from the physical to the virtual, and vice versa (see Section 5.2) i.e., dealing with the mixed reality of pervasive games. In Section 5.3, using advances from the domain of Virtual Worlds, all three architectures are compared and scaled up to distributed systems. PAPER VII (marked with the working title Delegation in Figure 1.2) explicates the problem of scaling architectures for virtual worlds (MMOWs) and IoT. In Section 5.4 (Iteration IV), results from PAPER VII are applied to pervasive games, extending pervasive games into a broader set of pervasive applications incorporating IoT. This leads to the future work presented in Chapter 8.
2. Research Strategies / Methods

Design Science is a special strand of design research, with the . . .

. . . “intent to produce and communicate knowledge that is of general interest. […] In contrast [to design], Design Science produces results that are relevant for a global practice i.e., a community of local practices, and for the research community. The different purposes of design and Design Science give rise to three additional requirements on Design Science research. Firstly, the purpose of creating new knowledge of general interest requires design science projects to make use of rigorous research methods. Secondly, the knowledge produced has to be related to an already existing knowledge base, in order to ensure that proposed results are both well founded and original. Thirdly, the new results should be communicated to both practitioners and researchers.”

– Johannesson and Perjons [2014, p.8].

“Design Science is the scientific study and creation of artefacts [sic] as they are developed and used by people with the goal of solving practical problems of general interest” [Johannesson and Perjons 2014, p.7, original italics] “In contrast to empirical research, design research is not content to just describe, explain, and predict. It also wants to change the world, to improve it, and to create new worlds. Design research does this by developing artefacts [sic] that can help people fulfil [sic] their needs, overcome their problems, and grasp new opportunities” [Johannesson and Perjons 2014, p.1]. Artifacts are defined here as objects made by humans to address a particular problem; this includes methods and guidelines. Each artifact has an inner structure that can produce certain behaviors i.e., offer function for people in a practice.

As methodological support for projects, Design Science offers researchers the Method Framework for Design Science Research, (see Figure 2.1). The method framework consists of five activities that range from problem investigation to demonstration and evaluation; any research strategy can be used in any of the five activities. “Many Design Science projects do not undertake all of the five activities of the method framework in depth. Instead, they may focus
on one or two of the activities, while the others are treated more lightly” [Johannesson and Perjons 2014, p.79]. Varied focus is the basis for at least five typical cases of research, namely: problem-focused; requirements-focused; requirements- and development-focused; development- and evaluation-focused; and, evaluation-focused Design Science research.

Figure 2.1: Method Framework for Design Science Research - An overview of the five activities in one iteration of the method framework [Johannesson and Perjons 2014, Fig 4.1, p.77, with permission of Springer Science+Business Media]; each activity is depicted as a square with input and output on each side.

2.1 This Dissertation is Design Science Research

This dissertation is requirements- and development-focused Design Science research for distributed technology-sustained pervasive applications, making use of four iterations (see Figure 1.2) of the Method Framework for Design Science research. A pervasive application is a socio-technical system i.e., a hybrid system consisting of both artifacts, as well as humans, and the influences that govern their actions. This dissertation is from a technological perspective, but design requirements must still take into account how humans affect the system and are affected. Because the solution maturity in the other domains (e.g., Computer Games or GIS) is high and the application domain maturity of pervasive games is low, this dissertation makes use of several exaptations.
2.2  Multiple Iterations of Method Framework

For each iteration of the method framework, a section is included below outlining the focus, the artifact and research strategy used.

2.2.1  Design and Development of a New Medium for a GDD

Iteration I of the method framework was development- and evaluation-focused Design Science research. The artifact for PAPER I, was a model of a system that could serve as a new medium for a Game Design Document (GDD); a communication tool for game designers, as well as form of documentation. Existing GDD mediums were not adequate for the game development community, explicit in previous work [Nevelsteen 2008]. Requirements analysis and initial design of the model were performed by this author through document survey i.e., reading published criticisms, game development ‘post-mortems’ [Dingsøyr 2005] and surveying existing technologies (see Section II of PAPER I). The model was a ‘sketch’ [Johannesson and Perjons 2014] of core functionality. The design would take much inspiration from existing technologies such as a wiki or the then current technology at the time, called Wave [Google 2009c]. Traveur, presented next, was the first project assigned during this author’s PhD studies and the GDD was a side project. Because development resources were to be spent on Traveur, it was decided that the GDD would not be implemented, but only modeled.

Design and evaluation of the model was done iteratively in three phases, in collaboration with and as supervisor for master student, Sergio Gayoso Fernández [2011]. Under the guidance of Annika Waern, formative evaluations at the end of each phase were done with observations in a controlled environment and phenomenological unstructured interviews of participants. As a final summative evaluation of the core functionality exhibited in the model, an interview with designers at a leading game development company was performed and compared to the model.

2.2.2  Traveur: Pervasive Architecture and Map Functionality

For Iteration II, this author focused on requirements- and development-focused research, while others in our research group focused on development and evaluation. The initial problem was presented to our research group by Parkour community stakeholders. The artifact for the Traveur project was the design and development of a pervasive game or service for the Parkour community i.e., an ‘instantiation’ [Johannesson and Perjons 2014, p.29] of a running fully functional prototype, comprised of smaller artifacts (e.g., the map function). Artifacts were demonstrated in ‘local practice’ [Johannesson and
Perjons 2014, p.77] and evaluated as case study. A team of two game designers and four developers would design and development Traveur. “The project went through four distinguishable phases”: brainstorming workshop, technological probing, prototyping and the implementation of a ‘fully functional’ prototype [PAPER II].

2.2.2.1 Four Distinguishable Phases

The design goals of the Traveur project were to create a pervasive Parkour game, satisfying stakeholders and the Parkour community. Initial brainstorming of phase one was done (February 2nd, 2010) in a focus group with all stakeholders: the Helsingborg-based Parkour and Freerunning group, called Air-Wipp, as user representative; Street Media 7, a small company developing mobile phone technology, as commercial stakeholder; and our team as research stakeholder. The collaboration created a wide and diverse scope for the design of the game, but in subsequent meetings, it became clear that Air-Wipp did not desire any game-like aspects to be developed. There was fear that the Parkour community would reject any design that had a competitive aspect (due to the communities’ strong emphasis on collaboration), and the incitement to interact with people in public, outside the Parkour community, was perceived as very low. The goal of our research group, to create a pervasive Parkour game, was merged with Air-Wipp’s goal for creating a Parkour academy function, essentially becoming a training service, that would make it fun to train in a safe way. At the time, no such training service was available, and only towards the end of Traveur did one competing application become available. In actuality, Traveur turned out to be a very ambitious and unique project, aiming to combine: mobile community functionality, Parkour training support, real-time location-based mapping functionality and context-awareness in the form of biometric data. Although the number of active people in the project was adequate, the project was only to last six months.

Because Parkour is a high impact sport, it was unclear what technology could be adopted without disturbing the practice. During the second phase of the project, ‘technology probes’ (i.e., “very simple one-function applications that Air-Wipp could use in their daily practice and evaluate for their usefulness” [PAPER II]) were built as ‘instrumental case studies’ [Creswell 2013] for early feedback. Extensive work on the technology probes was done by this author; examples of which include: a YouTube [Google 2009d] uploader app, a simple location-based map with ‘meet-up’ function, and a body harness containing accelerometers. The YouTube uploader was needed, because uploading videos to YouTube via mobile device was not evident or even blocked by some carriers at the time, to prevent overuse of mobile network bandwidth.
The meet-up function, allowed users to see where other users (who were simultaneously using the same meet-up probe) were in real-time. Evaluation of the meet-up probe by Parkour practitioners was considered successful; the probe was used as intended and also ‘appropriated’ [Johannesson and Perjons 2014, p.162] as a geo-located chat service and as a tool for playing a technology-supported game of chase. It was decided to extend the meet-up function and incorporate it into the client prototype (see Section 2.2.2.2). Since users of Traveur already carried with them a smartphone [Wikipedia 2016, Smartphone], with a variety of sensors (e.g., 3G, GPS, Bluetooth, accelerometers, vibration motor), enabling context-awareness [BOOK I], and Parkour and Freerunning is a high intensity sport, our team found it interesting to try and capture live biometric data (galvanic skin response and pulse were being considered by piggybacking on the Affective Health [Silvăşan et al. 2010] project). Prototype harnesses were created to secure a smartphone during the high intensity Parkour jumps and rolls (see an artists rendition of a Parkour vault in Figure 2.2). Accelerometer data (e.g., speed, impact and rotation) was collected, but never tied into the Traveur system. Considering Traveur was designed to be a training guide, an advanced feature was to analyze the accelerometer data to determine if a particular Parkour movement had been accomplished e.g., similar to [Márquez Segura et al. 2011].
In phase three, prototyping as a ‘collective case study’ [Creswell 2013] was done in two iterations [PAPER II]; subsequent requirement analysis and design, after each formative evaluation, was done through participatory action research by colleagues. Initial design of the architecture was done by Annika Waern, from our research group, and Joel Westerberg, from Street Media 7, with added input from this author. The first evaluation, by Air-Wipp together with their students in Helsingborg, uncovered a need to re-design Traveur, to move it even further away from its original goals of creating a pervasive game. The fully functional prototype was again evaluated in a public test with participants from both Stockholm Parkour Academy and Uppsala Parkour. An account of how the design of the pervasive Parkour game shifted towards a pervasive service can be found in PAPER II. A detailed account of the underlying architecture and how it was affected by the shift in the design is presented in Section 4.2.

The fourth and last phase of the Traveur project focused on ‘value conflicts uncovered’ during previous tests. Traveur was evaluated again and subsequently redesign, resulting in PAPER II.

2.2.2.2 Map Functionality

Initially, development of the mobile client and the Log-of-Everything were assigned to this author. Because Traveur turned out to be a highly ambitious project, a consultant was hired to develop the client, alleviating this author to implement the technology probes and the map functionality; design of the map functionality was done by this author, under supervision of Annika Waern. Initial map functionality grew out of an idea from this author, to use a service such as Find My Friends [Apple 2011] or Latitude [Google 2009b], to allow Parkour practitioners to find each other. The meet-up function allowed a number of users to temporally share their location with others in real-time and leave geo-located comments on the shared map. At the time, such location sharing applications were only then becoming widely accepted. The Glympse [2008] app implements the meet-up functionality today.

Considering the Traveur interface was already offering users a map, it was decided to extend the map functionality [PAPER II] (see Figure 2.3) to a location-based mobile platform supporting: areas of interest, training data, various media types (e.g., text, sound, image and video), historical traces, a ‘heat map’ [Wikipedia 2016, ‘Heat map’] and the meet-up function. This lead to the following list of requirements:

- a shared map showing areas of interest with various levels of detail;
- real-time response i.e., updating of data every second;
- robust handling of mobile network and GPS ‘uncertainty’ [BOOK I];
- resolving the trade-off between coarse GPS updates (i.e., not suitable for tracking walking or Parkour running) and detailed GPS updates (i.e., battery life and not being able to use the service in a moving vehicle).
- ability to add content to a shared map (i.e., crowd-sourcing of content), with new content disseminated to clients (e.g., in a heat map of activity);
- and, the display of historic data on the map e.g., showing movement traces over time.

Areas of interest would be geo-located points or polygonal areas of ongoing events or training locations (good or bad). To each area of interest, training data could be assigned showing what Parkour exercises could be performed there. By allowing for various media types to be assigned to areas of interest, location-based content could be crowd-sourced from the Parkour community. Historical data on the map would allow for traces of movement over time to be shown on the map; this would allow Parkour runners to see a trajectory through the city that others might have taken previously. The heat map showing areas with a high level of activity was designed for, but never implemented.

Figure 2.3: Two images of the map functionality napkin design - on the left, a circle of interest with various content contained in it, and on the right, an interface sketch of a perspective view, with spin gestures and buttons.
2.2.3 Pervasive Games Architecture: Pervasive MOO

Iteration III of the method framework was largest, with focus on requirements, development and evaluation of an architecture for a pervasive game, called Codename: Heroes. Requirements for CN:H were provided informally as a “long term pervasive game”, with a staging spanning months or years, and developed ‘in house’. A single game designer and two developers would design and develop CN:H under the supervision of Annika Waern, who also contended with design. The project was planned from September 2011 to mid-April (i.e., five months, plus debugging and trials), but ran through September 2012. CN:H was based on the LambdaMOO (MOO) engine, which was subsequently repurposed for the IoMOOT project in December (see Section 6.4.3). The artifact for Iteration III is an instantiation of a system capable of staging a pervasive game i.e., the same for publications BOOK I, PAPER V and PAPER VI. The focus and research strategy differ for each of the four subsections below, but the artifact is not repeated.

2.2.3.1 Survey of Pervasive Games and Technologies

Given the limited resources available to produce CN:H, the problem made explicit by the Traveur project (see Section 4.2) was that if CN:H was to be successful, an architecture was needed that could cater to the requirements of a pervasive game. In order to find a feature set describing a would-be pervasive games engine, a systematic review [Ampatzoglou and Stamelos 2010] was performed as a survey of existing pervasive game projects and technologies. Chapter 2 of BOOK I is that systematic review, serving as an informal requirements analysis for a would-be pervasive games engine. The survey spans two large collaborations that together span 10 years, two books pertaining to pervasive games and, top ranking (first 100+) keyword search results, on Google Scholar, after the year 2009, for pervasive game: ‘technology’, ‘architecture’ or ‘engine’. Complementary searches were performed to find references, citing previously found references, that were considered key to the study, or referenced by Nieuwdorp (2007), relevant to the definition of pervasive games.

2.2.3.2 Virtual World Engine Staging a Pervasive Game

The focus for Chapter 3 of BOOK I was development and evaluation; the feature set discussed above was implemented in a running system and evaluated for feasibility i.e., a case study was the chosen research strategy. Using the feature set from above, a virtual world engine was chosen as an architecture in the same ‘product line’ [Bass, Clements, and Kazman 2013] as a would-be pervasive games engine [BOOK I]; for the implementation of CN:H, a virtual
world engine would be exapted to stage the pervasive game *i.e.*, a solution from the domain of Computer Games exapted to the domain of Pervasive Games. The CN:H architecture was demonstrated in two separate stagings of the game, with evaluation of CN:H from a cultural perspective done by colleagues.

### 2.2.3.3 Integrated Spatiotemporal Conceptual Model

The complexity of dealing with mixed reality (*i.e.*, mapping the virtual to physical, and *vice versa*) was made explicit in the case study of CN:H. Because the domains of GIS and Pervasive Games are both based on the Earth’s geography, PAPER VI obtains an integrated spatiotemporal conceptual model through the exaptation of Peuquet’s Triad Representational Framework, combining it with Dix et al.’s three types of space and a notion of time. The integrated model is a small artifact in the larger CN:H project. The focus of PAPER VI is evaluation research verifying the model, using simulation *i.e.*, successful implementation.

### 2.2.3.4 A Definition for ‘Virtual World’

Another problem that was made explicit in the case study of CN:H, was that a working definition for ‘virtual world’ was lacking. Virtual world engines existed, but since no working definition of ‘virtual world’ existed, there was no list of properties an engine had to exhibit for it to be called a ‘virtual world engine’. The focus of PAPER V was to determine the requirements for an artifact construct in the form of a definition *i.e.*, the requirements for a virtual environment to be called a ‘virtual world’ by definition. The definition represented a small artifact construct, which could be used in combination with the virtual world engine, in the larger CN:H project. Because there is no consensus on properties of a virtual world coinciding with technology implementing a world, the definition was obtained using grounded theory [Johannesson and Perjons 2014] rather than a literature review *i.e.*, a definition could be formed from any possible criteria related to a virtual world, rather than being limited to definitions or criteria found in previous literature. Theoretical sampling was used to select technologies that refined the definition and this was continued until theoretical saturation was achieved (see Section 4 of PAPER V for details). The obtained definition was evaluated by comparing it with prominent existing definitions. To evaluate the obtained properties a form of ‘discriminant sampling’ [Creswell 2013] was used *i.e.*, selecting advanced contemporary technologies that were not in the study to see if the theory holds true for these additional samples, classifying advanced technologies, such as: a pseudo-persistent video game, a MANet, virtual reality, mixed reality, and the Metaverse. The effectiveness of the obtained definition was demonstrated by creating an ontology of virtual worlds.
2.2.4 Aligning Research of MMOWs with that of IoT

The focus of Iteration IV is problem-focused Design Science research, explicating the problem of designing and developing an architecture for distributed pervasive applications. Properties, related to the scaling of architectures, are gathered from the domain of MMOWs and IoT through a document survey i.e., research surveys were sought pertaining to virtual worlds, MMOWs, IoT, the Metaverse, and Wireless Sensor and Actuator Networks. Properties found in the domain of MMOWs are evaluated against those found in domain of IoT. The artifact for Iteration IV is an architecture for distributed technology-sustained pervasive applications, which can be designed and developed in future research (see Chapter 8).
3. Related Work

In BOOK I, a systematic review was performed to find the then current state of the art in pervasive game architectures. Several publications were cited suggesting frameworks and middleware for pervasive games (i.e., custom-built solutions, see Section 2.4.1 of the book), but the solutions are not compared with existing game engine technology (more ‘general-purpose’ solutions [Gregory 2014]), and only a few publications repurpose an existing game engine in their project e.g., Ambient Wood [Thompson et al. 2003] or ARQuake [Thomas et al. 2000]. If that review is extended to the day of this writing, then the conclusion remains the same i.e., still no wide-spread use of advances in existing game engine technology for pervasive games. Viana et al. [2014] present a systematic review of research on software engineering pervasive games; their study shows an increase in development in the area of pervasive games in general, but with areas such as "Reuse" and "Interoperability" being under-explored. Their review is said to include three papers on software system architectures (i.e.,[S37, S74, S83]) that, similar to other recent publications [Fischer et al. 2014] and [Pimenta et al. 2014], provide custom-built solutions, but do not compare with the large body of knowledge found in the domain of Computer Games. Pimenta et al. [2014] recognize the need for a pervasive games engine, but only relate to technologies in the domain of Pervasive Games (e.g., MUPE). If pervasive games, do not take into account the domain of Computer Video Games, then developers of pervasive games will be left continually reinventing existing technologies.

Paelke, Oppermann, and Reimann [2008] state that many pervasive games “implement their own custom game engine by tying together available standard components for distributed applications and filling the gaps with custom code”. They demonstrate that a web server can serve as a game engine when tied to other components. Although a viable solution, “tying together available standard components” does not provide any foresight into how an architecture will behave with respect to scalability. If the game is suddenly successful, how must the architecture change to handle an increased load due to the influx of additional users? Also, after repeatedly creating many solutions for various pervasive games, one might want to collect commonly used components into an engine i.e., find out which characteristics are important for pervasive games
and build an engine to accommodate them. A recent publication, by Valente, Feijó, and Prado Leite [2015], does this by providing a formal requirements analysis for pervasive mobile games.

ARQuake [Thomas et al. 2000] is a version of the original video game called Quake [id Software 1996], extended into a pervasive game. ARQuake was developed by repurposing the original Quake engine, as mentioned by Lewis and Jacobson [2002]. Quake and ARQuake are similar to Doom (presented in PAPER V) in the sense that they both do not have a persistent virtual world. Because a persistent virtual world was not required by the game, ARQuake could be implemented by repurposing a “graphics engine rather than a virtual world engine, with a Euclidean spatial representation, that could be directly overlaid onto the physical world” [BOOK I]. Without a persistent virtual world, ARQuake’s times of play (when the player is in the game world) can not be temporally expanded [BOOK I] i.e., “uncertain, ambiguous, and hard to define” [Montola, Stenros, and Waern 2009]. A recent survey, by Kasapakis and Gavalas [2015], aims to classify pervasive games into age generations based on technologies used. The survey is limited to 18 pervasive games e.g., not containing tabletop games, smart toys and trans-reality games, even though being mentioned as sub-genres in the articles’ related work. Kasapakis and Gavalas [2015] state that “the game engine organization model is largely dictated by the game scenario to be supported” and this author agrees. If a general-purpose pervasive games engine is to be created, commonalities between game technologies must be found e.g., support for a virtual persistent world. Each engine, in the domain of Computer Video Games, has advantages and disadvantages which must be considered when repurposing it as a would-be pervasive games engine e.g., the focus of a graphics engine is typically high definition graphics for a few players, with less focus on building a large expansive world. Kasapakis and Gavalas [2015] continues that the “current technological status favours [sic] always-on connectivity, hence, centralized models”; this is not entirely correct since, “always-on connectivity” relates to persistence, which can also be obtained through decentralized models.

In Ambient Wood [Thompson et al. 2003] a MUD-based virtual world engine (with support for persistence) was chosen to stage the game, in order to “to replicate and model the physical world so that interactions between devices in the physical environment would have corresponding interactions within the model” [BOOK I]. Ambient Wood was a client/server architecture, with a custom-built proxy object called MEAP [BOOK I] handling the various heterogeneous devices e.g., mobile handhelds as sensors or sound actuators. Although MEAP was custom-built, it was developed as a more general-purpose tool for heterogeneous devices [See 2001]. There is no mention in the literature of possibly reusing the architecture of Ambient Wood to stage addi-
tional pervasive games, or if the architecture could be scaled to support more players. The architecture of Ambient Wood was a distributed system of heterogeneous devices, with one centralized MUD game engine. Thompson et al. [2003] stated that they were extending their work across distributed MUD engines in 2003, with an interest in the “delegation of ownership [authority] to localised [sic] MUDs”.

In a recent publication, Laine and Sedano [2015] mention the novelty of the pervasive game (exergame) Othello as being distributed gameplay, enabling “players from anywhere in the world to play against each other”. The architecture, however, was client/server “causing a noticeable delay [latency]” in communication to due the geographical distance between client and server. Laine and Sedano [2015] note that “unlike MMORPGs [MMOWs with a RPG theme], it is not necessary to divide computing across multiple servers [for Othello] because there are few concurrent players”. In a recent publication by Kamarainen et al. [2014], the subject of cloud computing is discussed as a possible solution for pervasive and mobile computing, allowing the “end-user device to offload computation, storage, and the tasks of graphic rendering to the cloud”. Similar to Othello, Kamarainen et al. [2014] remark that latency is the “main challenge” for cloud gaming, with most interactive games requiring response times that only “local deployment scenarios” can deliver. As a solution, they “propose to use [a] hybrid and decentralized cloud computing infrastructure, which enables deploying game servers on hosts with close proximity to the mobile clients”. To exploit local resources, the Fun in Numbers (FiiN) platform (presented in BOOK I) features a distributed multi-tiered (i.e., four layered) large-scale architecture [Chatzigiannakis et al. 2011]. The FiiN architecture supports more than one game engine, with each engine being the “local authority for each physical game site” [Akribopoulos et al. 2009]. All game engines are coordinated by a centralized top-most layer. The bottom layers of the FiiN architecture enable support for ad-hoc networks and IoT. The problem with the FiiN architecture is that it is unclear exactly what types of pervasive games are supported. Pervasive games are defined in the FiiN publications as “games played in the physical space, indoors or outdoors, using mobile handheld devices, context-awareness, and in certain cases some degree of infrastructure and scripting”. Chatzigiannakis et al. [2011] make no distinction between technology-sustained or technology-supported pervasive games, even though technology-supported pervasive games do not necessarily require a game engine i.e., the infrastructure to enable them is very different. Akribopoulos et al. [2009, original italics] state that FiiN targets “mainly games that involve multiple players, rapid physical activity, gesturing, . . . and less storytelling-based games”, which could account for why game master interfaces are not present in the architecture [BOOK I].
Above, it is stated that a web server can function as a pervasive games engine. In an early publication, Broll et al. [2009] describe the Perci Framework, which connects mobile devices (via a proxy) to a web server, forming a pervasive service that leverages IoT e.g., mapping the virtual to the physical through digital devices. The web server provides a persistent virtual world and it is the “nature of persistence [that] is closely related to the nature of ubiquitous as understood in the Ubiquitous Computing research field” [Lankoski et al. 2004]. The Perci Framework can be used by games or other applications, and is in the domain of Ubiquitous Computing.
4. Design of Three Software System Architectures

From the year 2000 to around 2009, there was lots of activity in the research community to build pervasive games, using either custom-built solutions or now outdated engines [BOOK I] e.g., MUD or MUPE. Without the advantage of hindsight, creators of pervasive games could not generalize the requirements for pervasive games [Bass, Clements, and Kazman 2013]. Considering video games have existed for decades, with reusable game engines to drive them, this dissertation sets out to see if engine technology from the domain of Computer Video Games can be repurposed to stage technology-sustained pervasive games. Discovering this was done in practice through the design of three software system architectures: the GDD [PAPER I], Traveur [PAPER II], and CN:H [BOOK I]. In this chapter, one section below is dedicated to one iteration of the Method Framework for Design Science Research for each of the three architectures. (The fourth iteration is in Chapter 5.)

Iteration I was a design exercise on how to model a persistent communication technology (the GDD), of which the requirements analysis, design and ‘view’ concept, influenced subsequent iterations. The requirements analysis and design of a pervasive service (Traveur) from Iteration II influenced Iteration III, with concepts such as heterogeneous servers, context-awareness and mixed reality. Poor architectural design lead to broadening the search for subsequent technological solutions to include the domain of Computer Games. Iteration III, the largest iteration, provides a characteristic feature set for pervasive games (evaluated through CN:H) that aligns pervasive games research with that of virtual worlds (based on the persistence trait). The CN:H case study highlighted that no usable definition for a ‘virtual world’ existed and that an integrated model, mapping time and space from the physical to the virtual, and vice versa, was lacking. PAPER V provides the definition and PAPER VI provides an integrated spatiotemporal conceptual model.
4.1 A Persistent Communication Technology

Even the GDD, of PAPER I, exhibits some pervasive characteristics *e.g.*, temporal expansion. The requirement analysis for the GDD resulted in four communication-based requirements for the medium:

- collaborative user editing with enabling communication mechanisms;
- being readily available at all times to all users *e.g.*, web based;
- changes communicated to the users, with differentials; and,
- support for a variety of different discussion channels *e.g.*, real-time and non-real-time based on video, audio or text.

And, seven additional general requirements:

- a mechanism to allow for narrative linearity and linear printing;
- editor roles with an option to force edits to be approved by a lead editor;
- a familiar user-interface and intuitive interaction model;
- quick updating, with fast access and editing;
- many media/file types *e.g.*, audio, video, images, spreadsheets, *et cetera*;
- the ability to link relative information, with auto-linking;
- and revision control tracking, with a backup system.

Many of these requirements, such as a web-based interface with revision control and auto-linking, were common and in wide-spread use at the time of the GDD project, but others were not so common. Wave [Google 2009c] had already been current for two years and Docs [Google 2009a] was available, but not yet a mainstream service open to the public *i.e.*, collaborative user editing via the Internet was not mainstream, as it is at the time of this writing. The notion of a ‘view’ (*i.e.*, a subset of available data as a customized visualization for a particular user), contributed by the GDD study, is still not in wide-spread use as of this writing.

Using PAPER V, the modeled GDD medium can be categorized as a persistent communication technology (see Section 5.1.1) *i.e.*, a persistent environment, spanning one shared data space, where users can interact in real-time. And, it is these properties that influenced both Traveur and BOOK I. In essence, the GDD served as an exploratory case study for pervasive technology-sustained applications.

4.2 Design of a Pervasive Parkour Game; 
shifting requirements and mashed-up components

The second architectural case study was that of Traveur, which had major architectural differences in design when compared with the previous project. In Traveur, participatory action research was to blame for fluid project require-
ments, highlighting that, in a pervasive setting “heterogeneity is the norm”\(^1\) i.e., a ‘mash-up’ [Singhal et al. 2013] of technologies with distributed computing. In comparison, the GDD medium was designed to ‘readily available’, but only for those who were already behind a computer desk designing or developing a video game; not those with mobile devices out in the physical world. In contrast, Traveur would strive to be a pervasive service for the activity of Parkour running, with the mobile phone as an interface to a virtual world i.e., striving towards ‘ubiquitous availability’ and ‘ubiquity of access’ [BOOK I]. Out of the architectures studied in this dissertation, Traveur was the most context-aware, making use of a wide variety of location dependent and biometric data. This data was captured, through sensors in smartphones held or strapped to Parkour runners, and could be displayed through the map functionality.

4.2.1 Architectural Back-end

The initial architecture for Traveur was done by Annika Waern, from our research group, and Joel Westerberg, from Street Media 7, with added input from this author. The architectural discussion was straightforward until a decision was needed on where persistent data was to be stored. Since Street Media was building a more general community platform based on their proprietary back-end, ‘Streetside’, it was argued that Street Media needed to maintain control over the community data. However, the existing ‘Creator’ component, from our research group, which was to handle the game aspects of Traveur, already had its own database connectivity. Both Streetside and Creator had been created to be a stand-alone central components in a system, and neither had built-in functionality to replicate data to other servers. At this point, the decision was made to make Traveur a mash-up of existing services into one application, allowing both parties control over their segment of the data. Because neither of the two systems would have a complete set of the data, a component called the Log-of-Everything\(^2\) was to be created such that it aggregated the data from Streetside and Creator (see Figure 4.1 for the original design on whiteboard, and Figure 4.2 for a sketch thereof). Aggregated data could then be exported to other services. Due to lack of resources, the Log-of-Everything was never created.

\(^1\)It was Annika Waern who coined this phrase.

\(^2\)It was at this point that the architectural sketch of the back-end, with the redirection of data and the Log-of-Everything, raised an eye-brow of this author.
Figure 4.1: Drawing of the Traveur architecture - Initial design on whiteboard; the top two squares represent one or more client applications, each cylinder represents a database of information, and arrows show the flow of information (for clarity see the sketch in Figure 4.2).

Figure 4.2: Sketch of the Traveur architecture - See Figure 4.1 for a picture of the original whiteboard and a legend. The text in red are notes labeling who is responsible for what component e.g., for game state and rules component, interface was to be handled by Kalle, and scripting by both Annika and Kalle.
4.2.2 Architectural Front-end

Initial prototyping was done on Android [Google 2008] smartphones, but one of the first shifts of the design requirements meant moving to the iOS [Apple 2007] mobile platform. Android was initially chosen because it was an open platform, allowing for direct access to phone hardware and an easier app publishing procedure (Apple employs an approval process through which all apps are screened before being published, making the distribution of prototypes difficult). But, feedback came from the Parkour community that most practitioners were using iPhones running iOS; design of the mobile client for Traveur project became that of an iPhone app using the native development framework to access the phone’s hardware and sensors. The user interface for Traveur consisted of five different ‘tabs’:

- News, a rolling blog of all news and events in the community;
- People, simply a long list of all participants in the system;
- Skills, which represented the game functionality provided by Creator;
- Map, an augmented WebMap implementing the map functionality (see Section 2.2.2.2); and, lastly a
- User Profile, to control user login.

Initially, News, People and Map tabs were filled using data from Streetside, leaving the Skills tab to be filled by Creator. The User Profile tab did not exhibit proper functionality until later in development. Because content was being supplied by two different servers, this meant that a connection to each server needed to be maintained \textit{i.e.}, lack of connectivity to a server meant that those tabs the server was responsible for were blank. Because the set of user profiles logged in the community function was larger than that of the game (since login by the general public into the community, not participating in the game, was considered ok), authentication of user identity \textit{(i.e., control over the logins)} was delegated to Streetside, and deemed to be replicated to Creator. This changed the dependency graph so that Streetside handled all client communication and proxied data for Creator (see Figure 4.3) \textit{i.e.}, Streetside could stand alone, but Creator was dependent on Streetside to replicate user data. Streetside was dependent on Creator, but this dependency was actually the client’s dependency passed through Streetside \textit{e.g.}, if Creator failed, Streetside would simply not be able to provide the client with the required data, but Streetside would not suffer. Because of the chosen replication algorithm, the order in which the services were started and the stability of each machine became an issue.
4.2.3 Heterogeneity is the Norm

Because Traveur aimed to develop a service under fluid project requirements and on top of existing heterogeneous technology, problems could have been anticipated. Neither Streetside nor Creator was constructed to work together with another system e.g., both systems had their own database and internally assumed they had a complete ‘primary copy’ [Yahyavi and Kemme 2013] of the data. Synchronization of the two systems, to ensure consistency of replicated user data, was implemented poorly, resulting in large amounts of latency. Because two stakeholders wanted control over the data, the dependency graph was such that the phone client depended on two dedicated servers, with an additional dependency between them (see Figure 4.3, AFTER). These dependencies were particularly fragile in a mobile setting where ‘uncertainty’ [BOOK I] in network connectivity is prevalent. Creator supported scripting, which was useful to change the rules of the game on the fly, but this usefulness was limited to the Skills tab i.e., did not extend to Streetside or the client. In order to support context-awareness using biometric data (e.g., galvanic skin response), the architecture would have to support streaming data, but neither Streetside or Creator were built with streaming in mind.

Figure 4.3: Traveur’s change of the dependency graph - On the left (BEFORE): Streetside and Creator servers were standalone, with no user data shared between them. On the right (AFTER): all mobile client communication is with Streetside and user data is replicated between the two servers.
Essentially, Traveur fed into the next project the notion of how not to do the architectural design for a pervasive system. The next architecture to be built, for CN:H [BOOK I], was to have a significantly smaller development team and less resources, so choosing the appropriate technologies for it would be critical. This lead to broadening the search for technologies to include solutions from the domain of Computer Games i.e., not wasting time implementing critical functionality that already existed in other technologies. Also from Traveur, it was obvious that, since neither Streetside nor Creator were ‘mature’ [BOOK I] systems, it had crippled their reliability, and if an architecture for a pervasive game was to provide ubiquitous availability, reliability would be required [BOOK I].

4.3 Pervasive Game Architecture

For CN:H, this author assumed the role of systems architect. In order to avoid the mistakes made in Traveur and to be able to choose the appropriate technologies for CN:H, a good understanding of the requirements for a pervasive games engine would be needed. Although there are plenty of publications on pervasive games, it would seem the majority are from a cultural perspective, designing the game, rather than from a technological perspective, describing the technical requirements [BOOK I]. Many pervasive game projects have been undertaken in the years of Equator and IPerG, but literature lacks describing common overlapping characteristics of the technologies needed for pervasive games. This is understandable, because during the Equator and IPerG years, architects of those projects could not have effectively determined generalized properties of a pervasive games engine, because they did not have the advantage of hindsight i.e., it is far easier to determine requirements for an architecture in hindsight, rather than when faced with the problem [Bass, Clements, and Kazman 2013]. Brooks [1995] refers to this as “planning to throw one away; you will, anyhow”. That a literary survey was needed, seems to be corroborated by the fact that smaller similar surveys were being performed around the same time (see Section 2.4.1 of BOOK I). Chapter 2 of BOOK I is a systematic review into characteristic engine features that describe a would-be pervasive games engine. These features can be considered a set of informal requirements from which a set of formal requirements can be drawn. The assumption is made that, if a would-be pervasive games engine is to be general-purpose it should have support for the resulting characteristics, even though not all pervasive games will make use of them e.g., World Persistence, Game Master Intervention, and Reconfiguration, Authoring and Scripting in Run-Time.
In the survey of BOOK I, the following component feature set was obtained:

**Virtual Game World with World Persistence**: an instance with virtual elements (at least one of which being the player); a spatiotemporal game world with interacting entities, that overlaps both the virtual and the physical world; a world that continues to exist and develop internally even when there are no people interacting with it (persistence); and, ubiquitous availability through a reliable architecture.

**Shared Data Space(s) with Data Persistence**: a common shared data space, with coordinated communication to it; and data persistence in the event of a shutdown or system failure *i.e.*, fault tolerant and recoverable.

**Heterogeneous Devices and Systems**: support for non-standard input devices, comprised of sensors and actuators, that form an ‘interface’ [Nieuwdorp 2005] between the player and the game; resolution of interoperability issues through a device abstraction layer; and the use of service-oriented architectures, for the offering of services.

**Context-Awareness**: context information (*e.g.*, location, body orientation, available resources including network connectivity, proximity to surroundings or noise levels) is obtained through sensor enabled heterogeneous devices or service-oriented architectures; dealing with uncertainty in position localization or networking.

**Roles, Groups, Hierarchies, Permissions**: various roles for player and non-player characters, organized in groups or hierarchies; different permissions or privileged information for the various roles or organizations, perhaps through an entirely different interface to the game.

**Current and Historical Game State**: including semi-static player info, a view of the current internal game state (*e.g.*, through direct inspectable properties or through a specialized management interface), a historical perspective of the game state (*e.g.*, through the logging of event data for post-game analysis), or any meta-level game information, such as game master documentation.

**Game Master Intervention**: semi-automatic execution of the game through game master intervention in run-time (*e.g.*, by directly manipulating the internal game state or through specialized interfaces, that potentially translate massive amounts collected game data into a human consumable form); game master intervention can be provided by a service-oriented architecture.
Reconfiguration, Authoring and Scripting in Run-Time: in the pre-game phase (e.g., for location adaptability), that can be extended into the in-game phase; data-driven reconfiguration of software, hardware and devices; dynamic story and content through content generation in-game; changing of the game rules through data-definition or run-time languages; autonomous agents; or the simulation of events for the Wizard of Oz [Dow et al. 2005] technique (see Section 2.3.8 of BOOK I).

Bidirectional Diegetic\(^1\) and Non-Diegetic Communication: through various channels and/or interfaces.

Many of these features are quite generic (e.g., Current and Historical Game State) and so are supported by engines in the domain of Computer Video Games. Features more specific to pervasive games are Heterogeneous Devices and Systems; Context-Awareness; Game Master Intervention; and Reconfiguration, Authoring and Scripting in Run-Time; and Bidirectional Diegetic and Non-Diegetic Communication i.e., non-diegetic communication is required in a virtual world, but not as pronounced as in a pervasive game where it is needed to cope with social expansion.

From Souza e Silva and Sutko [2009], it is obtained that pervasive games are persistent worlds (i.e., persistent in the physical world [Lankoski et al. 2004]), sharing the trait of a persistence with virtual worlds [Nevelsteen 2016b]. This means that it could be beneficial “to exploit the coextensive virtual world as a ‘behind the scenes’ resource for coordinating and managing devices and interaction in the physical space” [Greenhalgh et al. 2001] (see Figure 4.4). Because the feature set above includes a Virtual Game World with World Persistence, and a Shared Data Space with Data Persistence, a virtual world engine was chosen as being in the same ‘product line’ [Bass, Clements, and Kazman 2013] as a would-be pervasive games engine [BOOK I]. Many of the surveyed games in BOOK I featured a persistent virtual world, but not all e.g., ARQuake. If virtual world persistence is not needed (only physical), a different engine might be sufficient to stage the pervasive game e.g., ARQuake used a graphics engine [Thomas et al. 2000]. It is assumed that the would-be pervasive games engine must support virtual persistence to be more general-purpose. One looming problem remained; pervasive games do share the trait of a persistence with virtual worlds, but since there was no usable definition of a ‘virtual world’ [PAPER V], how was it possible to be sure that the architecture for a would-be pervasive games engine does indeed overlap with that

\(^1\)The ‘diegesis’ of a story consists of whatever is true in that story. Diegetic elements are ‘in the story’; non-diegetic elements are not.” [Bergström 2011, original italics]
of a virtual world? The case study in Chapter 3 of BOOK I evaluated that indeed a virtual world engine can implement a pervasive game, but it could be beneficial to know to what extent other traits are shared.

Figure 4.4: Coextensive worlds - A depiction of the virtual (Spirit) world co-extensive with the physical world; when the player logs in, the virtual world overlaid on top of the physical world becomes a mixed world. Items from the virtual world, such as the depicted scroll, are mapped to the physical world. [Lankoski et al. 2004, Figure 2], DOI: 10.1145/1028014.1028083 Copyright Association for Computing Machinery, Inc. Reprinted with permission.
5. Mixed Reality and Scalability

Using the definition for ‘virtual world’ from PAPER V and the integrated spatiotemporal conceptual model from PAPER VI, this chapter properly aligns pervasive games research with that of virtual worlds (rather than just on the persistence trait), and handles the mixed reality aspect of pervasive games. Architectures for pervasive games are scaled up to distributed systems using advances in architecture from the domain of Virtual Worlds. And, considering the use of non-standard input devices in pervasive games and the rise of IoT, the problem of extending pervasive games into pervasive applications, incorporating IoT, is explicated as Iteration IV of the method framework.

5.1 ‘Virtual World’ Defined and Applied

In PAPER V, a virtual world is defined as having the following properties: Shared Temporality (1T, requiring Virtual Temporality [VT]), Real-time (Rt), Shared Spatiality (1S, requiring Virtual Spatiality [VS] and 1Sh), One Shard\(^1\) (1Sh), Many Software Agents (SA, requiring VT), Virtual Interaction (Ix, requiring SA, 1S, 1T and by consequence 1Sh), non-Pausable (nZ, requiring Rt), Persistence (P, requiring world persistence [wP] and data persistence [dP]) and Many Avatar (Av). This led to the definition for a virtual world being:

A simulated environment where: MANY\(^2\) agents can virtually interact with each other, act and react to things, phenomena and the environment; agents can be ZERO\(^3\) or MANY human(s), each represented by MANY entities called a ‘virtual self’ (an avatar), or MANY software agents; all action/reaction/interaction must happen in a real-time shared spatiotemporal non-pausable virtual environment; the environment may consist of many data spaces, but the collection of data spaces should constitute a shared data space, ONE\(^4\) persistent shard.

---

\(^1\) “With shards, players are divided up into groups and assigned to a unique copy (a shard) of virtual space, with each shard handled by a different group of servers; players are prohibited from moving between shards” [PAPER V].

\(^2\) one or more.

\(^3\) exactly zero.

\(^4\) one and only one.
Using this definition, it is possible to verify if each of the software system architectures in Chapter 4 can support a virtual world. This can give an indication of how one architecture has built upon knowledge of the previous.

5.1.1 Is the GDD a Virtual World?

The GDD of PAPER I, was designed as a web-based technology with similarities to a wiki. The GDD fails to support Virtual Temporality (any recorded times would be related to Simulated Temporality) and so fails to support Shared Temporality. The GDD was designed to be readily available at all times for easy editing (i.e., not turn- or tick-based) and therefore a Real-time technology. Because the GDD was designed as a document style tool, it fails to support Virtual Spatiality and, by consequence, Shared Spatiality. The node structure of the GDD was designed to be possibly highly distributed, but still One Shard. No software agents were designed for and therefore Virtual Interaction was only between users i.e., a communication technology. Since events were recorded according to Simulated Temporality, the non-Pausable criterium is easily filled. World Persistence is supported through the requirement for the GDD to be readily available at all times, albeit for designers of games behind a computer. Data persistence is an explicit requirement of the GDD, using revision control tracking and a backup system. Each user in the system would be given an entity to which multiple views and nodes could be linked, satisfying the Many Avatar property. The result for $\forall p: \land$ is then negative: $\not\vdash VT[1T,SA,Ix] \land VS[1S]$ (see Table 5.1) i.e., failure to provide a shared virtual spatiotemporal environment for interaction with software agents. Rather than a virtual world, the GDD can be classified as equal to Google Docs [PAPER I], a persistent communication technology.

5.1.2 Is Traveur a Virtual World?

From Table 5.1 it can be seen that Traveur scores similarly to the GDD with respect to being a virtual world; only satisfying one more property. Traveur also recorded times according to Simulated Temporality, failing to support Shared Temporality i.e., since Traveur was a mash-up of technologies, it was easier to rely on real-world time than to synchronize a virtual time. Traveur was a Real-time technology, with the map functionality showing updates in the range of seconds. According the characteristics of BOOK I, it is not required that all pervasive games make use of Virtual Spatiality, but through the map functionality, Traveur supported Virtual Spatiality that was mapped to the physical world. Although the synchronization between the Streetside and Creator components was poorly implemented, they did form a shared data space i.e., One Shard. Software agents were never implemented in Traveur; doing so would
have been a reason to implement Virtual Temporality. Similar to the GDD, software agents were never implemented, so Traveur also fails to support Virtual Interaction. Simulated Temporality was used, rather than Virtual Temporality, so the non-Pausable property was trivial to fulfill. World Persistence was supported through the mobile phones providing ubiquity of access for the Parkour runners; although poorly designed, the architecture strived to maintain ubiquitous availability. Each of both Streetside and Creator had their own persistent database, satisfying the Data Persistence requirement. And, finally, each user in the system was assigned an entity in the Streetside mobile community, which was then used system wide i.e., an avatar. The result for $\forall p: \land$ is negative: $\models \forall T[1T, SA, Ix]$, (see Table 5.1). In comparison to the GDD, Shared Temporality is implemented in the map functionality, but the lack of Virtual Interaction leaves Traveur insufficiently worldly.

5.1.3 Pervasive Games Overlap with Virtual Worlds?

PAPER V determines a set of properties that virtual worlds exhibit. If an engine features those properties, then a virtual world is supported by the engine i.e., formal requirements can be formed on how to enable those features in the engine. The choice of using a virtual world engine as a would-be pervasive games engine can be revisited by comparing the properties of a ‘virtual world’ (in Section 5.1) with the desired characteristic feature set for a pervasive games engine (in Section 4.3) i.e., it is possible to see what other properties they share besides persistence.

Shared Temporality and Spatiality were explicitly named in the feature of a Virtual Game World with World Persistence; Virtual Temporality is implied, because Many Software Agents and Persistence are also required (see further). Nothing dictates that all pervasive games must be Real-time, but if a would-be pervasive games engine is to be more general-purpose, it must support Real-time. A pervasive game world is said to overlap with both the virtual and physical, classifying a pervasive game as mixed reality according to PAPER V i.e., a pervasive game has Shared Spatiality, but not necessarily virtual. The part of the game world player is accessing is not necessarily the virtual one;

<table>
<thead>
<tr>
<th>TECHNOLOGY</th>
<th>1T</th>
<th>Rt</th>
<th>1S</th>
<th>1Sh</th>
<th>SA</th>
<th>Ix</th>
<th>nZ</th>
<th>P</th>
<th>Av</th>
<th>$\forall p: \land$</th>
</tr>
</thead>
<tbody>
<tr>
<td>the GDD</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Traveur</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>pervasive engine</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>*1</td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 5.1: Classification of the architectures according to PAPER V

---

1$Ix$ requires $1S$, which must be satisfied by physical, virtual or mixed spatiality.
a distinction between pervasive games and pervasive computing “lies in relation to ubiquity of access; pervasive computing without access to a computing device might prove difficult, but a player partially denied access to a computing device can potentially still be in the game” [BOOK I]. The property of One Shard is explicit in the characteristic of a Shared Data Space with Data Persistence and the property of Many Software Agents is supported through the characteristic of Reconfiguration, Authoring and Scripting in Run-time. Interaction is mentioned in a Virtual Game World with World Persistence, but Virtual Interaction requires virtual Shared Spatiality, which is not a requirement for a pervasive game i.e., a technology-sustained pervasive game is mixed reality. If only physical spatiality is used in the pervasive game, Virtual Interaction is possible for those physical entities that have virtual counterparts and share physical spatial proximity. If both physical and virtual spatiality are used, Virtual Interaction is possible between virtual entities, including the virtual counterparts of physical entities (if those entities share physical or virtual spatial proximity). Physical entities with virtual counterparts require a mapping from the physical to the virtual, and possibly vice versa. The property of non-Pausable is not mentioned explicitly as a characteristic in the feature set, but is implied through world persistence and ubiquitous availability. Persistence is mentioned explicitly in the form of both World and Data persistence. And, finally, if players of the pervasive game are to act and react to the virtual world, they must have an entity “through which all their in-world activity is channelled” [PAPER V] i.e., an avatar. The result for ∀p:∧ is then negative: X ⊨ VS[1S] (see Table 5.1) i.e., a pervasive game is not guaranteed to support a virtual world due to the games’s mixed spatiality.

An initial reaction to the negative result might be to conclude non-correlation (i.e., pervasive games characteristics do not fully describe a virtual world), but this result clearly shows that the majority of properties defining a virtual world are characteristics that are preferable to a would-be pervasive games engine i.e., a substantial improvement over choosing a virtual world engine as being in the same product line as a would-be pervasive games engine, based solely on the overlapping trait of persistence [BOOK I]. From the result it is possible to conclude that, because of the mixed nature of a pervasive game, not all pervasive games must support a virtual world, but they can. Considering the definition of ‘virtual world’, the characteristic of a Virtual Game World with World Persistence of BOOK I should then be read as: a game world where some elements are virtual i.e., not a virtual world per se. As an example, Traveur was supposed to be a pervasive game; Traveur could feature a virtual world, but isn’t required to do so; if a virtual world is featured, the would-be pervasive games engine should be able to stage it.
5.1.4 One World or a World of Worlds?

In Section 5.3 the architecture for pervasive games will be scaled up to accommodate a massive amount of interactions. PAPER VII compares issues from the domain of MMOWs (large scale virtual worlds) with those from IoT, to explicate the problem of extending pervasive games into a broader set of distributed pervasive applications, incorporating IoT. It is a tautology that a MMOW supports a virtual world, but it can be questioned whether architectures, for distributed pervasive applications incorporating IoT, implement one virtual world or many. A contribution of PAPER V is that it clearly defines what constitutes a single virtual world or multiple, in the face of distributed (centralized and decentralized) systems. The property of One Shard dictates that “if a technology has more than one data space, those spaces must be merged at one point or another for them to be considered one virtual world; each other copy/shard is considered another world” [PAPER V]. It is still to early to know if IoT will be enabled by a single architecture or middleware, but there is risk of ‘fragmentation’ [PAPER V]; it is this fragmentation that coincides with a negating of the One Shard property, and in all likelihood the property of Shared Temporality as well.

5.2 Mapping Between the Virtual and Physical

The overlap between pervasive games and virtual worlds is made clear in Section 5.1.3; a pervasive games engine does not necessarily fully support a virtual world. It was particularly a lack of Shared Spatiality that caused a negative result; to what degree a pervasive game makes use of the physical or virtual world can vary, but a characteristic of a technology-sustained pervasive game is that the game world overlaps with both the virtual and the physical [BOOK I] i.e., a pervasive game is mixed reality. How then is a pervasive games engine supposed to handle this type of spatiality? In Section 5.1.3, pervasive games are said to support Virtual Temporality, but temporality can also be mixed (see Section 5.3.5 in PAPER V). How is this to be handled? In the domain of Pervasive Games, research exists tying the virtual to the physical, but an “integrated spatiotemporal conceptual model for mapping the virtual to the physical and vice versa” [PAPER VI] is lacking. Because Geographical Information Systems and Pervasive Games both make use of the Earth’s geography, their problem domains overlap i.e., research found in GIS, on how to model the physical world, can be exapted to Pervasive Games. The contribution of PAPER VI is that it provides an integrated conceptual model through the exaptation of Peuquet’s Triad Representational Framework [1994], combining it with Dix et al.’s three types of space [2005] and a notion of time. The notions of time
by Langran [1992], and Zagal and Mateas [2007], from the domain of GIS and Pervasive Games respectively, are equated and incorporated.

Considering an overall conceptual representation for geographic phenomena, it can be assumed that any such representation is composed of entities, properties and relationships [Peuquet 1988]. In a cartographic representation, two dominant views exist, the ‘geometric structure’ (e.g., size, shape, orientation, color, height and location) and the ‘graphic image’ (e.g., temperature or height of a particular location or relation to other locations). In geometric structure, the entity is a spatial object (‘object-based’), whereas in the graphic image, the entity is a location (‘location-based’). Spatial objects are higher-order information, based on information from a location-based perception of the physical world. Although not entirely distinct, Peuquet presents these views in a unified Dual Model [1988], on the account that “neither view is intrinsically better than the other, but are logical duals of each other”. If the object-based view is referred to as the WHAT, and the location-based view as the WHERE, then using the Dual Model, two categories of spatial queries can be formed:

- WHAT → WHERE e.g., given an object, where is it located?
- WHERE → WHAT e.g., given a location, what objects are located there?

Object-Based Representation  
According to Wachowicz [1999], “any relation defined on a set of entities creates a space . . . defining a relation automatically defines a space” i.e., virtual objects of a pervasive game (the WHAT) and the relations between them constitute a space. Two spatial relations, that can be used to organize the WHAT of a pervasive game, are a taxonomic hierarchy or a set of objects. The taxonomy allows for the objects to be organized according to their ‘inheritance’ [Peuquet 1994] of attributes or thematics i.e., thematic modeling [Abdul-Rahman and Pilouk 2008].

Location-Based Representation  
For pervasive games, that make use of positioning technology, a single georeferenced point constitutes an object in the location-based representation of the WHERE [Peuquet 1994], with several georeferenced points forming a bounded area also adhering to this representation. The spatial relations applicable to pervasive games are metric and topological; spherical distance with the 9-intersection model [Zlatanova, Rahman, and Shi 2004; Abdul-Rahman and Pilouk 2008] being sufficient to describe the organization of two objects. A directed graph can be constructed in a pervasive game (forming a finite state machine e.g., for quests) by having location objects serve as nodes and orientational from-to relations (directed edges) between nodes [Abdul-Rahman and Pilouk 2008].
Dix et al. [2005] have shown how virtual space can be mapped to and from physical space through ‘measured space’, a representation of space captured in sensors (and actuators – projection is given as an example of mapping directly from virtual space to the physical, but this author would argue that such a projection would also have to cross measured space e.g., the projector also requires calibration). Combining Peuquet’s Dual Model and Dix et al.’s three types of space, virtual objects (spatial objects in the WHAT) can be related to virtual locations (locations in the WHERE) and mapped indirectly (through measured space) to physical locations, and vice versa. If geometric structure is read by sensors, this can be mapped similarly to a virtual object.

Since pervasive games can be played in physical ‘real-world’ time [Zagal and Mateas 2007] (equal to ‘world time’ [Langran 1992, p.34] in the domain of GIS), change must be accounted for. Peuquet extends her own Dual Model with a time-based view, forming the Triad Representational Framework [1994]. In the time-based view, the basic entity is a single unit of time. Using Triad, virtual time (time in the time-based view) can be related to virtual objects and virtual locations. If the time-based view is referred to as the WHEN, then Triad enables spatiotemporal queries of the following forms:

- WHAT + WHERE → WHEN e.g., given an object at a particular location, when was the last time it changed or appeared?
- WHAT + WHEN → WHERE e.g., given an object in a time span, what trajectory through space did the object take?
- WHERE + WHEN → WHAT e.g., if at a particular location, what objects passed by after a particular time?

**Time-Based Representation**  
In a pervasive game, a single unit of ‘virtual time’ [PAPER V] (assuming VT is supported, else ST) is the basic entity in the WHEN. According to Peuquet [1994], all temporal relationships can be divided into three distinct classes: (1) metrics and topology; (2) boolean operators; and (3) generalization. Only the first class is needed for pervasive games in this article: temporal distance being “the length of the interval between any two given locations along a time-line” and a topology defining how two temporal events relate [Peuquet 1994] (also see Langran [1992] for details).

Virtual time (‘gameworld time’ by Zagal and Mateas [2007]) can be considered an abstraction of time simulated by a computer system [PAPER V]. If virtual time in the Triad framework is instrumented to and from physical time, then virtual time can be indirectly mapped to physical time (and vice versa) over ‘measured time’, similar to the technique for physical space. Langran [1992, p.34] notes the difference between physical time and measured
time, with respect to GIS literature, as “the difference between when events occur in the world and when the database records them”.

In the game engine, virtual objects of the WHAT are related to the WHERE and WHEN. Each of the three views of Triad can be implemented using the overall conceptual representation of an entity with properties, connected to other entities via a relation. A more compact representation is possible, if an entity has location-based and time-based views stored directly in its properties. This completes the conceptual model for mapping the physical to the virtual and vice versa i.e., handling the mixed reality of pervasive games. With the domain of Pervasive Games properly aligned with the domain of Virtual Worlds (and mixed reality dealt with), this then also completes the design of a would-be pervasive games engine using a virtual world engine.

5.3 Scaling up the Architectures to Distributed Systems

By aligning the domain of Pervasive Games with that of Virtual Worlds, the architectures from Chapter 4 can be scaled up to distributed systems using advances in architecture from the domain of Virtual Worlds.

5.3.1 Client / Server

A requirements analysis was performed for the new GDD medium and a model created, but no concrete system was architected. The most straightforward architecture to implement the GDD model would be that of client/server (see Figure 5.1); clients connecting to a single server through HTTP, with one or more redundant servers as backup (redundant servers are not shown in Figure 5.1). Data persistence would be through the use of a database, and elementary interoperability through the specification of HTTP, as the communication protocol; HTML can play an important role in allowing the client to use dynamic content and interfaces [BOOK I]. The architecture of the GDD bares resemblance to that of CN:H (compare Figure 5.1 with that of Figure 5.2), but without a sensory system on the client, for context-awareness, or a proxy component supporting heterogeneous systems (outside of HTTP for interoperability). Since all transactions could be relayed through a centralized server, providing authority would be trivial. Scalability is problematic with a single server [Yahyavi and Kemme 2013], handing perhaps hundreds or thousands of simultaneous users, but not more. Cloud computing can be used in this scenario, but this is the discussion for the next section. The backup server must be kept up to date in the event of system failure, but consistency is trivial, since the active server always has the primary copy [Yahyavi and Kemme 2013].
Figure 5.1: **GDD architecture** - One centralized GDD component (with a redundant backup) serving one or more desktop computers.

Figure 5.2: **CN:H architecture** - The MOO game engine component serving mobile devices with sensors, and a GM (game master) interface tool through a proxy object. This figure is a replication of Figure 3.2 of BOOK I; a detailed explanation of the architecture can be found in Section 3.3 of the book.
If a sensory system were to be added to the GDD architecture, the architecture would start to resemble that of CN:H. The sensory system can be in the form of, for example, mobile phones with sensor hardware, wireless sensor networks, or service-oriented architectures providing external sensor data. If many types of sensors are used simultaneously, support for heterogeneous devices and systems must be provided for; in the CN:H architecture, the proxy component (depicted in Figure 5.2) partly handled interoperability.

5.3.2 A Centralized Server Cluster

Although the MOO game engine, used in the CN:H architecture (see Figure 5.2), fulfilled many of the required characteristics as a would-be pervasive games engine, the component was outdated [BOOK I]. Since the domain of Pervasive Games has been aligned with that of Virtual Worlds, advances in virtual world architecture can be used to scale up pervasive games architectures to distributed engines. Virtual world engines are already in existence that use a centralized cluster of servers for load-balancing (e.g., BigWorld [2002]) [BOOK I], so the MOO game engine component in Figure 5.2 can be replaced with a centralized cluster of servers to achieve cloud computing. This to an extent alleviates the scalability problem, but in exchange for communication overhead needed to coordinate the cluster [Yahyavi and Kemme 2013]. To lower latency and bandwidth, improving scalability further, a hybrid solution can be considered i.e., a peer-to-peer system in combination with a centralized server cluster (see Section 5.3.5).

5.3.3 Heterogeneous Servers

Figure 5.3 depicts the final Traveur architecture, after the change of the dependency graph, from two direct dependencies to one direct and one indirect dependency (see Figure 4.3 for the prior graph). To make the heterogeneous servers Streetside and Creator interoperable, each would have to behave as a service-oriented architecture. Streetside did provide REST [Fielding and Taylor 2000] interfaces and Creator supported modular adaptors to mitigate communication, but the services were not robust. The initial dependency graph of Traveur (see Figure 4.3, BEFORE) depicts each client having a direct connection to Streetside and Creator. In this scenario, user data was redundant and inconsistent in each of the two components and different parts of the same Traveur service could be unavailable depending on which server was down. To resolve the problem of inconsistent redundant user data, Creator could query Streetside for login authentication, but an odd situation could still arise where the user was successfully logged into Creator (via Streetside), but with Streetside’s part of Traveur becoming unavailable shortly after login. And, since
Streetside and Creator were not in the same geolocation, querying from one server to the other incurred latency. Changing the dependency graph (see Figure 4.3), with the client only needing a single connection to Streetside, resolved the authentication situation so that user data could be replicated and the entire Traveur service could be deemed out of service. This did not resolve the client’s dependency on Creator, which was residing on another hardware server.

![Diagram of Traveur architecture, after change of dependency graph](image)

**Figure 5.3: Traveur architecture, after change of dependency graph** - The community function (which included user login data) was accessible through both mobile device and desktop computer, through HTTP and/or REST.

Since both Streetside and Creator were designed to be stand-alone servers, replication services were not considered at design time. Consistency was implemented through a bulk transfer of changes (to user objects) to Creator, but since Streetside and Creator were not in the same geolocation, this transfer incurred large amounts of latency. A more efficient replication algorithm for user data would have to be implemented, but development resources in Traveur were already overused. An obvious solution would be to colocate both servers in the same geolocation, but this was denied by the stakeholders.

Considering Traveur had heterogeneity server side, the architecture was more complex than that of CN:H; in the face of this complexity, a few different approaches could have been taken. First off, since Streetside and Creator both provided a unique part of the Traveur service, ideally both should have ensured availability. As mentioned in BOOK I, this requires “reliability
i.e., mature systems, preventing failure by being fault tolerant and supporting recoverability”. Neither Streetside nor Creator were mature systems; each had been under development and not fully tested for stability (engine maturity was a primary reason for choosing a MUD-based engine for CN:H). When errors were encountered in Streetside or Creator, these caused system down time rather than less severe levels of failure i.e., Traveur should have provided graceful degradation. Also, in the case of system failure, no redundant backup servers were in place to maintain the service. Since the Traveur client was directly or indirectly dependent on both components, the chances of having a degraded system were higher than if a single server were used. As a solution, if only two physical hardware servers were available, two redundant copies of the same component could have been run on their respective servers, with a dynamic switch on failure. Or, as an alternative, a monitoring process could have been started to monitor each critical component for downtime, restarting the component in the event of failure. In the case of hardware failure, this still means a degraded system, since the redundancy and monitoring would be affected by the failure. Running a copy of each component on each server hardware is an alternative, but this was denied due to the fact that each stakeholder wanted control over their own service. If both Streetside and Creator had been built as replicable modular instances that scaled, cloud computing could have been leveraged in the form of IaaS [Wikipedia 2016, Infrastructure as a service], allowing for scalability and control over each instance.

Although not encountered at the time, Traveur suffers the same scalability problem as CN:H i.e., not scalable servers supporting a limited amount of users. If both clients and servers are heterogenous, then a peer-to-peer system or a hybrid architecture could have been considered (see Section 5.3.5).

### 5.3.4 Heterogeneous Clients

Traveur highlighted that in a pervasive setting, “heterogeneity is the norm”, stemming from the situation that the server components used in Traveur were heterogeneous. In actuality, the situation in Traveur could have still been worse. The Streetside community pages were designed to be available on a wide range of devices via HTTP and HTML, and the client was switched from Android OS to iOS, but only one single smartphone type was fully supported simultaneously i.e., non-HTTP clients could also have been heterogeneous leading to more interoperability issues [BOOK I]. Heterogeneity of peers can be found in peer-to-peer networking e.g., peers can be chosen to be ‘superpeers’, supporting higher responsibilities and with higher levels of resources [Yahyavi and Kemme 2013]. In general high degrees of heterogeneity client side means more interoperability issues.
5.3.5 Peer-to-Peer

As mentioned in Section 5.3.2, scalability can be improved by moving towards a peer-to-peer system or a hybrid architecture [Yahyavi and Kemme 2013] i.e., allowing clients to communicate directly with each other rather than having to communicate indirectly over the server (see Figure 5.4). But, care must be taken to consider security and cheating, so that clients don’t interrupt information dissemination, perform illegal actions, or gain access to unauthorized information [Yahyavi and Kemme 2013]. It is possible to move away from a centralized server cluster entirely (e.g., distributing the authority of the game), but security and cheating in such a scenario remain areas of open research [BOOK I]. Yahyavi and Kemme [2013] state that, also persistence in decentralized architectures for games “has not received a lot of attention”. The survey of Chapter 2 in BOOK I includes the Transhumance platform, a decentralized architecture for the pervasive game called Team Exploration. In a mobile setting, however, devices have limitations in connectivity, latency, memory, and processing power. Yahyavi and Kemme [2013] state that this makes it “unlikely that mobile platforms will become powerful peers in a massively multiplayer game” (e.g., a MMOW that is a game [PAPER V]), but do add that peer-to-peer mechanisms might be advantageous in a local setting (see Section 5.4.2).
5.4 Distributed Pervasive Applications, Incorporating IoT

Pervasive games need a sensory system for context-awareness (which is usually made up of heterogeneous devices [BOOK I]); part of IoT could serve as such a sensory system i.e., pervasive games will also have to contend with IoT. Considering game engines can be repurposed for pervasive games [BOOK I], it is possible to talk about a broader set of pervasive applications, enabled by pervasive games engines with support for IoT.

With the rise of the Internet of Things, research is ongoing to design and develop a platform that can enable IoT [PAPER VII]. Research in the domain of large scale Virtual Worlds (i.e., MMOWs) includes various distributed architectures to deal with the issue of scalability, having already encountered issues that are now also surfacing in the domain of IoT. PAPER VII surveys the domain of MMOWs for properties that are affected by scaling an architecture, and then evaluates how these are dealt with in the domain of IoT, noting the similarities and dissimilarities.

In PAPER VII, six properties related to scaling have been discussed, with one conclusion being that “MMOWs can clearly learn from advances, in resource utilization, availability, and responsiveness with respect to (partially) disconnected networks, from the domain of IoT”. Since the domain of Pervasive Games overlaps with the domain of Virtual Worlds (i.e., MMOWs, see Section 5.1.3 and PAPER V), pervasive games can also learn from advances in those properties of scale. In the following sections, the problem, of extending pervasive games into a broader set of pervasive applications incorporating IoT, is explicated. Explicating the problem for distributed technology-sustained pervasive applications, is the beginning of Iteration IV of the method framework; the rest of the iteration is left as future work (see Chapter 8).

5.4.1 Resource Utilization

One of the main tenets of IoT is the support for sensors and mobile devices having limited computing power i.e., ‘smart objects’ [López et al. 2011; Miorandi et al. 2012]. Support for smart objects is not common in the domain of MMOWs, so research on such objects can’t be directly adopted by pervasive games from the domain of Virtual Worlds, but the domain of Pervasive Games does have its own research with respect to Heterogeneous Devices and Systems (see Section 4.3) e.g., devices (with sensors and actuators) often enable the virtual to be mapped to the physical and vice versa. Service-oriented architectures is an approach being used in the domain of IoT to support interoperability between various platforms and systems [Atzori, Iera, and Morabito 2010; Miorandi et al. 2012], which is also part of Heterogeneous Devices and Systems.
for pervasive games. Interoperability is an open issue in both IoT [Gubbi et al. 2013; Miorandi et al. 2012; Atzori, Iera, and Morabito 2010] and Pervasive Games [BOOK I]. López et al. [2011] argue that IoT shall “aim to provide an architecture that will make use of the strengths of both SOA [service-oriented architectures] and agent-based systems”; if pervasive games adopt an architecture from the domain of Virtual Worlds, then agents are already supported (see Section 5.1).

5.4.2 Availability

The domain of MMOWs has a considerable amount of research towards peer-to-peer and hybrid solutions [PAPER VII], but rarely contends with (partially) disconnected networks. Because IoT must contend with (partially) disconnected networks [López et al. 2011], pervasive games more closely relate to IoT than MMOWs in this respect. In the event of a disconnected state, remote resources (e.g., cloud computing) are often unavailable, but local resources might still be available, depending on the architecture e.g., if the authority over local resources is a remote centralized one, then authorization over local resources cannot be obtained in a disconnected state and access to local resources is denied. For MMOWs, a centralized architecture is often preferred giving the game developer more control over the architecture [Yahyavi and Kemme 2013], but if pervasive games are to support (partially) disconnected networks, this will require the “rethinking of game platforms and game engines”\(^1\). In the domain of Computer Games, this rethinking or ‘revisiting’ is corroborated by Messaoudi, Simon, and Ksentini [2015].

PAPER VII states that “for consistency and security, there seems to be advances in both domains that can cross over to the other domain”; it is precisely in a decentralized scenario where there is much activity in the domain of IoT that can possibly be adopted by MMOWs, and by extension pervasive games.

5.4.3 Responsiveness

“To achieve the appropriate responsiveness, IoT is said to require two classes of traffic: the throughput and delay tolerant elastic traffic; and, the bandwidth and delay sensitive inelastic (real-time) traffic” [PAPER VII]. Considering real-time data “is one of the properties that distinguishes MMOWs from other distributed systems” [PAPER VII], pervasive games relate more closely to IoT with respect to responsiveness e.g., in BOOK I, the use of ‘delay-tolerant’ networking is mentioned as a solution in partially disconnected networks.

\(^1\)as Theo Kanter always says.
6. Evaluation

In the previous two chapters, it has been shown how work on the GDD and Traveur influenced the design and development of a would-be pervasive games engine. Research on pervasive games has been aligned with that of virtual worlds and the mixed reality of pervasive games dealt with. Pervasive games have been scaled up to distributed architectures using research on MMOWs. And, finally, research from IoT was used to explicate the problem for future research on distributed pervasive applications, incorporating IoT.

In this chapter, the research question and methodology of the dissertation is first evaluated, followed by an evaluation of the work of Chapter 4 and 5.

6.1 Validating the Research Question

The entire research question follows from the primary question of: can engine technology from the domain of Computer Video Games be repurposed to stage technology-sustained pervasive games? That engine technology from domain of Computer Video Games can be repurposed has already been established in literature [Lewis and Jacobson 2002]. Before expanding to pervasive applications, the domain of this dissertation is limited to Technology-Sustained Pervasive Games; computer video games are also technology-sustained and so share this trait with the games herein. The similarity of the two domains is exemplified by the statement by Montola, Stenros, and Waern [2009] that technology-sustained pervasive games are “computer games interfacing with the physical world”.

If the shared activity of gaming is looked at, a similar overlap can be observed. In the Pervasive Discourse by Nieuwdorp [2007], she explains that the first time the word ‘pervasive’ was used in conjunction with ‘gaming’ was in relation to LARP (live action role-play), of which MUD was considered a ‘virtual counterpart’. MUD is a virtual world in which a game can be played.

6.2 Verifying the Research Methodology

In order to qualify as Design Science research, instead of Systems Development, projects must fulfill the three requirements stated verbosely in the be-
ginning of Chapter 2. They can be summarized as: (i) projects make use of rigorous research methods; (ii) the knowledge produced has to be related to an already existing knowledge base; and, (iii) new results should be communicated to both practitioners and researchers. An outline of the research methodology and strategies can also be found in Chapter 2. The knowledge produced in this dissertation is related to the existing knowledge base presented in Chapter 3 and the related work of each individual publication included in this dissertation. Results have been communicated to the research community through publication and to practitioners through research collaboration e.g., Street Media publicizing the Traveur project (see Section 2.2.2) and CN:H being staged in collaboration with a Stockholm youth festival.

6.3 Evaluating the Design of the Architectures

Each publication contains its own evaluation, but the analysis of how each system is said to build on previous iterations must be evaluated. A characteristic feature set for a would-be pervasive games engine was derived in BOOK I and verbosely summarized in Section 4.3. In the following three subsections, the three architectures of Chapter 4 are evaluated against the feature set.

6.3.1 The GDD as a Basis for Pervasive Applications

To evaluate the GDD with respect to being a pervasive application, the requirements of the GDD (in Section 4.1) can be compared with the desired characteristic features for pervasive games i.e., highlighting the pervasiveness of the GDD, even though it is not a game (see Section 2.5.2 of BOOK I). With respect to the target audience for the GDD (individuals working with desktop class computers), the virtual environment was required to be “readily available at all times to all users”, which is exactly the requirement for World Persistence [BOOK I]. In order to support “collaborative user editing”, users must have access to the same Shared Data Space. The specified revision control and backup system for the GDD provides the necessary Data Persistence for the data space and also provides a Current and Historical (Game) State for the system. The GDD then successfully offers ubiquitous availability to a persistent virtual world for the target audience i.e., support for temporal expansion. A web-based approach was suggested for the GDD, so as to be able to support users on Heterogeneous Devices and Systems i.e., providing ubiquity of access to the target audience. The GDD did not have any form of Context-Awareness to qualify as a pervasive system (see Section 2.5.1 of BOOK I). To handle social expansion, the GDD was required to feature Roles and Permissions for
editors and a Bidirectional Communication for both user communication and update dissemination.

A contribution of the GDD which is still not widely adopted, as of this writing, is the ‘view’ concept. The view returns in CN:H as a custom interface for each participant role in a pervasive game (see Section 2.3.5 of BOOK 1) and leading to an open issue (see Section 4.1.4 of BOOK 1). The GDD proves to be a persistent communication technology (see Section 5.1.1), the properties of which where used in the definition for a ‘virtual world’ in PAPER V.

6.3.2 The Heterogeneity of a Pervasive Service

The GDD was not pervasive (only supporting temporal expansion for the target audience) and so Traveur was the first pervasive project that was done during this author’s PhD studies. Before shifting to a pervasive service, Traveur was a game and so the architecture can be evaluated as the basis for that of CN:H i.e., which features were inherited by CN:H and contributed to the characteristic feature set.

To a certain degree Traveur supported all features of the set: A Virtual Game World was provided for, but World Persistence was poor due to instability of the system. Data Persistence was not an issue, there was no loss of data, but the algorithm ensuring consistency of the Shared Data Space was slow, incurring latency in the system. Traveur as a whole was only available on a single platform, the iOS mobile platform (i.e., there was little support for Heterogeneous Devices), but the HTML interface provided by Streetside was with the intent of supporting a wide variety of devices. Traveur was ambitious with regard to Context-Awareness, making use of a wide variety of location dependent and biometric data. Support for Roles, Hierarchies and Game Master Intervention is evident in the training system employed by Traveur; trainers with special privileges would be the control mechanism, to make sure trainees had completed the proper training, before moving on to the next level. Current and Historical Game State was available in both Traveur as a whole and in the map functionality. Runtime Authoring and Scripting was provided for by Creator on the Skills tab (see Section 4.2.3), but Streetside did not offer any support for scripting at the time, since it was still under development. The Traveur client served as a unidirectional Diegetic Communication channel, in the sense that the community function, provided by Streetside, showed the activity of other participants in real-time. The ability to leave comments in the map function was also appropriated as a type of slow Bidirectional Diegetic Communication channel. Although not an explicit part of the Traveur service, Traveur ran on a smartphone i.e., the phone itself could used for Bidirectional Communication.
The design of the map functionality for Traveur (see Section 2.2.2.2) fed into the client, server and WebMap tool, for CN:H e.g., dealing with position localization, real-time updates, uncertainty, and the visualization of game state on a WebMap. The concept of “heterogeneity being the norm” influenced BOOK I, stimulating research into the challenges surrounding interoperability. That two stakeholders in the project disputed over control of their segment of the data, fed into PAPER VII the notion of authority.

6.3.3 Evaluating the Feature Set for Pervasive Games

The CN:H architecture, in Chapter 3 of BOOK I, is used to evaluate the characteristic feature set as case study. CN:H did support all of the feature set, however: the architecture could have been decentralized to stress test the Shared Data Space concept, the use of Context-Awareness could have been more pronounced and non-diegetic communication could have been developed.

6.4 Evaluating the Combining of Results

In this section, the work combining publications from Chapter 5, handling the overlap of worlds, mixed reality and scalability, are evaluated.

6.4.1 Overlap of Pervasive Games and Virtual Worlds

Since no usable definition existed for a ‘virtual world’, in PAPER V, properties of a virtual world were obtained through grounded theory, and used to form a definition. In Section 5.1.3, the obtained definition is used to determine if pervasive games support a virtual world. The overlap is such that all properties of a virtual world are shared by a pervasive game, except for the mixed reality aspect of pervasive games. To verify that the properties of a virtual world (summarized in Section 5.1) are actually supported, the CN:H architecture (used in the case study of BOOK I) can be examined to see if all properties are indeed supported.

**Shared Temporality (1T)** The CN:H architecture makes use of the MOO game engine, a MUD-based engine (see Chapter 3 in BOOK I). According to PAPER V, in order to support Shared Temporality, an architecture must support Virtual Temporality. MOO offers access to Simulated Temporality and Virtual Temporality; a programmer is able to create ‘timers’ [Hess 2003] for software agents which are stored as offsets from an epoch of zero [PAPER V]. Considering CN:H used a centralized architecture, creating Shared Temporality was trivial; timers were created centrally in the game engine and the output
disseminated to all clients. CN:H was temporality expanded [BOOK I] and Simulated Temporality was used to record when certain events had entered the system e.g., last known player position or when a skill had been performed. No attempt was made to double check the accuracy of Simulated Temporality; it was assumed clocks on individual devices were synchronized using Coordinated Universal Time. Since CN:H made use of both Simulated and Virtual Temporality, CN:H is temporal mixed reality.

Real-time (Rt) All devices were able to communicate with the MOO engine in real-time. The MOO architecture was built to handle multiple concurrent connections, via the Telnet protocol, in a Virtual Terminal program [BOOK I]. In CN:H, Telnet in combination with MUD Client Protocol was utilized for all connecting devices; clients connected directly to MOO and were responsible for keeping the connection alive for real-time communication. To enable the WebMap game master tool, a proxy object (see Figure 5.2) was used to keep the connection alive, since the WebMap tool itself opened one or more connections on demand instead of a constant connection.

Shared Spatiality (1S) Because pervasive games are spatially expanded, making use of mixed reality, pervasive games (i.e., CN:H) are not required to support Virtual Spatiality (see Section 5.1.3). The notion of space in MUD comes from narrative and a room-based topology [PAPER V]. In CN:H, the room-based topology was repurposed for the questing system, and a GPS-based coordinate system used as a spatial representation instead. CN:H maintained a virtual representation of physical space, but players were not made aware of any notion of space beyond the physical i.e., the client directed them through physical space, without the use of a visual map or similar. But, the WebMap tool did provide a real-time visual map of Virtual Spatiality for game masters.

One Shard (1Sh) Similar to the Shared Temporality property, this property is trivial due to the centralized architecture of MOO; all clients can cache data, but the authority and primary copy holder of all objects is the single server. Clients can not function without a valid connection to the server.

Many Software Agents (SA) To support Many Software Agents, an architecture needs to support Virtual Temporality [PAPER V]; which CN:H does, see above. In the MOO engine, all objects have the potential to be software agents through scripting. A primary example of this is the void_walker non-player character, which mimicked a real CN:H player [BOOK I].
Virtual Interaction (Ix) If Many Software Agents are to have Virtual Interaction they must share virtual proximity, requiring both Shared Spatiality and Shared Temporality i.e., they must share at least one abstraction of time and One Shard [PAPER V]. CN:H satisfies the One Shard property and supports both Shared Simulated and Virtual Temporalities i.e., abstractions of time. In order for a technology to support Virtual Interaction, Shared Spatiality is also required i.e., users must be able to interact with both other users and act/react to the world [PAPER V]. CN:H made use of Virtual Spatiality that was overlaid on the physical through the use of GPS. All players had a virtual counterpart in the game, through which they could interact with other players (or the void_walker, which mimicked a player) and the world. To act/react with the world, players could act/react to physical objects and their virtual counterparts, through the game client and QR Codes [Denso Wave 1994], as long as the player and object shared physical (and by consequence virtual) proximity (see Section 5.1.3). As diegetic communication channels, in CN:H, players were able to interact with each other through message passing and proximity. But, “using the client as an non-diegetic bi-directional communication channel was decided against, as not to break the mythos of the game” [BOOK I]. No non-diegetic communication was created in CN:H, so this defaulted to email and phone conversations i.e., interaction, but not virtual.

non-Pausable (nZ) The non-Pausable property requires the Real-time property [PAPER V]. CN:H was real-time and temporally expanded, so the game world was available to any player at any time i.e., the virtual game world could not be paused jeopardizing availability.

Persistence (P) The Persistence property requires an architecture to support both World and Data Persistence. The underlying characteristic for persistence is based on the work by Richard Bartle, cocreator of the original MUD1 [PAPER V] i.e., stating that MUD supports persistence is a tautology. CN:H is a MUD-based architecture and supports both World and Data Persistence.

Many Avatar (Av) Each user of CN:H (including game masters) was assigned a player entity, as per the MOO engine, through which their virtual actions/interactions were channeled. Because the MOO engine asserted that each user was to only have a single identity, the MOO engine did not support crossmedia [BOOK I]. To allow game masters to be logged in to the system via two different devices, simultaneously, a ‘ghost’ player was created to maintain the second logon [BOOK I].
By evaluating the CN:H architecture against the properties of a virtual world, the result is the same as shown for a pervasive game in Table 5.1; leaving the mixed reality of a pervasive game to be evaluated.

6.4.2 Evaluating the Virtual / Physical Mapping

In Section 5.2 it has been shown that, through the exaptation of Peuquet’s Triad Representational Framework combined with Dix et al.’s three types of space and a notion of time, physical time and space can be mapped to virtual time and space, and vice versa. PAPER VI evaluates the integrated spatiotemporal conceptual model, using CN:H as a case study.

![Diagram of TRIAD Model](image)

**Figure 6.1:** Integrated spatiotemporal conceptual model applied to CN:H

In CN:H, the WHAT consists of game objects (organized in the taxonomic MOO hierarchy), as well as various groups of objects administered by a specialized generic_admin_group. Objects in the MOO hierarchy and groups (the WHAT) were either created during the development of the game (e.g., the game_master profile), or in run-time (e.g., the virtual counterparts, of crowdsourced player-generated game artifacts). The WHEN consists of snapshots of events at regular intervals and the WHERE of GPS measurements [PAPER VI]. In the MOO engine, all three views are related to each other (see the Information Model in Figure 6.1), by having the WHEN and WHERE stored directly in the properties of the WHAT. The game engine holds the primary copy of entities, with ‘replicas’ [Yahyavi and Kemme 2013] being disseminated.
When a player starts their game client for the first time, they are asked to create a profile. By creating a profile, a virtual player object is created and placed in the MOO hierarchy. After profile creation, the player’s location can already be read from the player’s device; sensors on the player’s device pick up GPS signals, measuring longitude, latitude and possibly time [NCO 2013]. The instrumented location measured by the sensors, constitutes measured space (see Figure 6.1). Measurements saved to device storage (with possible error at this stage or subsequent stages of storage, due to limitations of internal representation, transmission, rounding or truncation), constitute virtual space in the information models (the WHERE). Although it is possible to obtain a snapshot of time from the GPS signal [NCO 2013], in CN:H, a timestamp was formed using device time, which was (by default) synchronized with Coordinated Universal Time. Instrumented device times constitute measured time. Time snapshots (possibly different for each device) are transmitted to the server and reconciled with the game engine’s Simulated Temporality, constituting time in the information model (the WHEN). In hindsight, a shortcoming of the CN:H implementation was that timestamps were stored as Simulated Temporality and not as Virtual Temporality, although the MOO engine supported it. To create a player-generated game artifact, a player was required to register the artifact with the game client, causing a virtual object (a counterpart in the WHAT) and corresponding QR Code to be created. Players could then print the QR Code and attach it to a physical object i.e., the physical object gains a virtual counterpart. If a player wanted to act/react with the physical object, they first had to share physical proximity (close enough to scan the QR Code), at which point the location of the physical object was made equal to the location of the player i.e., virtual proximity.

The WebMap game master interface prototype of CN:H visualized player positions in the game world, using Current and Historical Game State i.e., GPS traces could be drawn on the interface map. The WebMap connected simultaneously with both the game engine and OpenStreetMap [OSMF 2004]; the game engine for game data and OpenStreetMap for map data. Replicas from all three views of the WHAT, WHEN and WHERE are transmitted via networking to the system hosting the WebMap. Since the WebMap can not be physically misaligned (e.g., like sensors or a robotic arm), this author would argue that the person reading the display and checking the physical space is the instrument (possibly misaligned with the physical world), constituting measured space and time e.g., not finding the player where expected. And, of course, it is not possible that measured space and time are exact when compared to the infinite resolution of physical space and time.

The integrated spatiotemporal conceptual model of PAPER VI highlights redundancy or dependencies in an architecture. This is evaluated using the
Traveur architecture. User profiles of Traveur are stored in Streetside along with their properties and replicated to Creator, so that each profile can be linked to skills in Creator, that can be performed at a particular location. Although related to location and time, without Triad, user profiles would be left an an implementation detail. Using Triad, user profiles represent the \textit{WHAT} and the redundancy through replication is highlighted. The \textit{WHERE} is location data generated through the map functionality, on the client side, and networked to the Streetside community server. When data is recorded or changed, the time of the event is recorded, representing the \textit{WHEN}. A spatiotemporal query of the form:

- \textbf{WHAT} + \textbf{WHERE} $\rightarrow$ \textbf{WHEN}:  \\
  “when was the last time a trainer with the jump skill was here?”

will cause, the \textbf{WHAT} (all trainers) to be queried from Streetside, the \textbf{WHAT} and corresponding property (whether they have obtained the jump skill) to be reconciled with Creator, the \textbf{WHERE} of trainers and the \textbf{WHEN} (in present and the past) to be queried from Streetside. If results are replicated to client devices, objects and properties remain in their respective Triad representations.

The evaluation of the integrated spatiotemporal conceptual model in combination with the previous Section 6.4.1, means the overlap of pervasive games with virtual worlds and also the mixed reality of pervasive games has been evaluated.

6.4.3 Scaling Up Architectures and Incorporating IoT

The alignment of the domain of Pervasive Games with the domain of Virtual Worlds allows advances from the domain of Virtual Worlds to be exapted. In Section 5.3, it was shown how the architectures could be scaled using advances in the domain of Virtual Worlds. These advances are grounded in existing research, in that section.

PAPER VII surveys the domain of MMOWs (\textit{i.e.}, large scale virtual worlds) for properties that are affected by scaling an architecture, and then evaluates how these are dealt with in the domain of IoT. Since the domain of Pervasive Games is aligned with that of Virtual Worlds, in Section 5.4, PAPER VII is used to explicate the problem of pervasive games having to contend with scale and incorporate IoT. In the beginning of Section 5.4 it is stated that part of IoT could potentially be used as the sensory system for pervasive games. Considering game engines can be repurposed [BOOK I], it is possible to talk about a broader set of pervasive applications.

To evaluate the broader set of technology-sustained pervasive applications, a prototype dubbed IoMOOT (Internet of MOO Things) was built using the CN:H engine, and subsequently demonstrated at VIP Day 2012, hosted by
this author’s research group at the time. A small scale model of a residential home was used as the physical space for the demonstration. The virtual spatial representation of MUD was used to simulate the space of the residential home. Inside the model home, various ‘things’ were present (e.g., a lamp, fan and thermometer), each of which with a virtual entity in the CN:H engine. The user of the system was also given a virtual entity in the engine, moving around virtual space relative to movement in physical space; the position of the user was simulated rather than using indoor position localization. The device abstraction layer provided by MUD Client Protocol [BOOK I], made it possible to easily replace the two user interfaces and provide connectivity to the IoT enabled home. The game client of CN:H on iOS was replaced by a TouchOSC [hexler.net 2009] interface, and the WebMap, which served as game master interface, was replaced by a web page displaying the status of the home e.g., lights on/off, fan on/off and current temperature. Via mobile device a user could use the TouchOSC interface to control the devices in the home remotely. On any web enabled device or system, the status of the home could be checked. IoMOOT allowed the user “to exploit the coextensive virtual world as a ‘behind the scenes’ resource for coordinating and managing devices and interaction in the physical space” [Greenhalgh et al. 2001]; this time applied to IoT rather than pervasive games.
Currently, there are no reusable game engines available for pervasive games and without such engines, developers are left continually reinventing existing technologies. If technology-sustained pervasive games can be understood as computer games interfacing with the physical world, the first part of the research question of this dissertation asks can engine technology from the domain of Computer Video Games be repurposed to stage technology-sustained pervasive games?

Several related works suggest frameworks and middleware for pervasive games (i.e., custom-built solutions), but in those works the solutions are not compared with existing game engine technology (more ‘general-purpose’ solutions [Gregory 2014]) and only a few works repurpose an existing game engine. Paelke, Oppermann, and Reimann [2008] state that many pervasive games “implement their own custom game engine by tying together available standard components for distributed applications and filling the gaps with custom code”. Although a viable solution, “tying together available standard components” does not provide any foresight into how the architecture will behave with respect to scalability.

ARQuake was developed by repurposing the original Quake engine, as mentioned by Lewis and Jacobson [2002]. Because a persistent virtual world was not required by the game, ARQuake could be implemented by repurposing a “graphics engine rather than a virtual world engine. But, without a persistent virtual world, ARQuake’s times of play (when the player is in the game world) can not be temporally expanded. In Ambient Wood [Thompson et al. 2003] a MUD-based virtual world engine (with support for persistence) was chosen to stage the game. There is no mention in the literature of possibly reusing the architecture of Ambient Wood to stage additional pervasive games, or if the architecture could be scaled to support more players. Kasapakis and Gavalas [2015] state that “the game engine organization model is largely dictated by the game scenario to be supported” and this author agrees. If a general-purpose pervasive games engine is to be created, commonalities between game technologies must be found.

This dissertation contains the design and development of three software system architectures and clearly shows how each influenced the requirements for technology-sustained pervasive games (see Chapter 4). The GDD, in PAPER I, proved to model a persistent communication technology, of which the
requirements analysis, design and ‘view’ concept (see Section 4.1) influenced Traveur and CN:H. Properties of a persistent communication technology influenced the definition of a ‘virtual world’ in PAPER V (see Section 5.1.1). Traveur highlighted that “heterogeneity is the norm” and served as exploratory research into pervasive and context-aware computing. A detailed account of the Traveur architecture, and its connection to PAPER II, has been presented in Section 4.2, the requirements analysis and design of which influenced CN:H. BOOK I contains an extensive systematic review into pervasive games, resulting in a feature set describing characteristic features of a would-be pervasive games engine. Using the feature set, a virtual world engine was chosen as being in the same product line as a would-be pervasive games engine, based on the shared trait of a persistence (see Section 4.3).

A contribution of BOOK I is that it highlighted that no usable definition for ‘virtual world’ existed and a model, mapping time and space from the physical to the virtual, and vice versa, was lacking. PAPER V provides a definition for a virtual world and it is used to show that the domain of Pervasive Games overlaps with that of Virtual Worlds, except for a discrepancy based on mixed reality (see Section 5.1); the majority of properties defining a virtual world are characteristics that are preferable to a would-be pervasive games engine. This leaves only the mixed reality of a pervasive game to be dealt with. PAPER VI provides a integrated spatiotemporal conceptual model through the exaptation of Peuquet’s Triad Representational Framework, combining it with Dix et al.’s three types of space and a notion of time (see Section 5.2).

Since computer game engines can be used to stage a pervasive game, the research question continues by questioning whether advances from the domain of Computer Video Games can be used to scale up pervasive games to distributed systems. It has been shown (in Section 5.1 and 5.2) that domain of Pervasive Games does overlap with that of Virtual Worlds. By aligning the domains advances in architecture, from the domain of Virtual Worlds, are used to scale up pervasive games architectures to distributed systems (see Section 5.3). In a recent publication by Kamarainen et al. [2014], the subject of cloud computing is discussed as a possible solution for pervasive and mobile computing. They remark that latency is the “main challenge” for cloud gaming, with most interactive games requiring response times that only “local deployment scenarios” can deliver. As a solution, they “propose to use [a] hybrid and decentralized cloud computing infrastructure, which enables deploying game servers on hosts with close proximity to the mobile clients”.

The second part of the research question asks: considering the use of non-standard input devices in pervasive games and the rise of Internet of Things, how will this affect the architectures supporting the broader set of pervasive applications? To exploit local resources, the Fun in Numbers (FiiN) platform
(presented in BOOK I) features a distributed multi-tiered (i.e., four layered) large-scale architecture [Chatzigiannakis et al. 2011]. The bottom layers of the FiiN architecture enable support for ad-hoc networks and IoT. In the domain of IoT, “endeavors already underway in an attempt to create an IoT platform, but a solution that addresses all the aspects required by the IoT is yet to be designed” [PAPER VII]. The contribution of PAPER VII is that it surveys the domain of MMOWs for properties that are affected by scaling an architecture, and then evaluates how these are dealt with in the domain of IoT. Using results from PAPER VII, pervasive games (the domain of which overlaps with MMOWs, see Section 5.1.3 and PAPER V) are extended to pervasive applications, incorporating IoT i.e., properties pertaining to scalability from the domain of MMOWs and IoT are compared and the result used to explicate the problem of scaling architectures for pervasive applications (see Section 5.4). PAPER VII contains open issues for MMOWs and IoT, which extend to pervasive games. In an early publication, Broll et al. [2009] describe the Perci Framework, which connects mobile devices (via a proxy) to a web server, forming a pervasive service that leverages IoT. The Perci Framework can be used by games or other applications, and is in the domain of Ubiquitous Computing. The open issues and possible overlap between the domain of Pervasive Games and Ubiquitous Computing is listed as future work in Chapter 8.

Evaluation of the dissertation is done in Chapter 6. Evaluating the design and development of the three software system architectures is done in Section 6.3; Chapter 3 of BOOK I is the evaluation of the characteristic feature set and the choice of a virtual engine as pervasive engine, verified through the case study of CN:H. The combined results of this dissertation are evaluated in Section 6.4; PAPER VI evaluates that the integrated spatiotemporal conceptual model is applicable to pervasive games. And, advances from the domain of Virtual Worlds, used to scale up pervasive games to distributed systems, are grounded in existing research.

The aim of this dissertation has been to allow advances from the domain of Computer Video Games to be used to scale up technology-sustained pervasive games to distributed systems. And, considering game engines can be repurposed, incorporating IoT, explicate the problem for distributed technology-sustained pervasive applications. The implication of this dissertation is to ensure that pervasive games are not left reinventing existing technologies.
8. Future Work

The most obvious future work, is the design and development of an architecture for distributed pervasive applications. Chapter 4 of BOOK I outlines open issues for a would-be pervasive games engine. And, PAPER VII and Section 5.4 explicit the problem of scaling such an architecture and incorporating IoT. Taking that work into account, the following points then summarize the direct extensions of this dissertation:

- In BOOK I, distributed and decentralized architectures are already listed as a challenge for pervasive games engines. Such architectures have the potential to improve scalability, but security and cheating are far more problematic than for a centralized architecture. Ad-hoc networks are also touched upon in BOOK I, but the concept is generalized to (partially) disconnected networks in PAPER VII. If resources are to be available and accessible in a disconnected state, then this will require the rethinking or revisiting of game platforms and game engines e.g., with respect to authority, consensus and consistency.

- The extension of ubiquitous computing is mentioned in BOOK I and sometimes as a vision for IoT [Gubbi et al. 2013]. IoT has the potential to offer more context-awareness through sensor technology and more control of the physical environment through actuators. The complexity of dealing with a massive amount of context-information in combination with the existing entities and agents in a game engine must be dealt with; this opens up avenues of research in the areas of Ambient Intelligence [Davies, Callaghan, and Alghazzawi 2014; Xiao 2015] and Big Data [Zaslavsky, Perera, and Georgakopoulos 2012].

- Rehm, Goel, and Crespi [2015] state that the “rather intangible idea of the IoT […] has recently been replaced by the broader, more tangible, concept of Cyber-Physical Systems”. Similar to how a virtual world is used as a ‘behind the scenes’ resource for pervasive applications in the dissertation, Rehm, Goel, and Crespi [2015] “believe that virtual worlds can serve as platforms to facilitate the integration required by Cyber-Physical Systems”. This new domain boasts feedback systems
that integrate computation, networking, and physical processes, through embedded systems \textit{i.e.}, real-time computing.

- If decentralized architectures are to succeed, mechanisms must be found to provide for security and privacy \textit{e.g.}, authority and data integrity. In BOOK I, anti-cheating is mentioned, but in IoT this is generalized as security and privacy.

- Interoperability is already mentioned as an issue in BOOK I, but the incorporation of IoT makes the issue even more important. Pervasive games are usually under the control of a limited number of stakeholders, so interfacing between heterogeneous systems could remain manageable. PAPER VII mentions the possibility that IoT will fragment into many different platforms. Contrary to pervasive games, if IoT fragments, then accessing IoT will require the interfacing with many platforms, including those with the various classes of responsiveness mentioned in PAPER VII (see Section 5.4.3).

- In this dissertation, domain of Pervasive Games has been aligned with that of Virtual Worlds. Pervasive game architectures have been scaled up to distributed systems, and the problem of scalability has been explicated for distributed pervasive applications, incorporating IoT (see Section 5.4). Considering the extension of ubiquitous computing is mentioned in BOOK I and sometimes as a vision for IoT, the domain of Pervasive and Ubiquitous Computing should be surveyed so that the architectures there can be compared to those of distributed pervasive applications and IoT.

- If IoT does fragment into many different platforms, then the fragmented platform structure plus technology-sustained pervasive applications, aligned with virtual worlds, resembles the concept of the Metaverse [Dionisio, Burns III, and Gilbert 2013] (which includes pervasive concepts such as augmented reality). There are various definitions of the Metaverse [Frey et al. 2008; Rehm, Goel, and Crespi 2015], but a fragmented platform structure resembles the system of interconnected virtual worlds, described by Frey et al. [2008] \textit{i.e.}, an internet of virtual worlds.
**Technologies**


— (2009d). *YouTube*. URL: [https://www.youtube.com/](https://www.youtube.com/).


Broll, Gregor et al. (2009). “Perci: Pervasive service interaction with the Internet of Things.” In: *Internet Computing, IEEE* 13.6, pp. 74–81. DOI: 10.1109/MIC.2009.120.


