GRANODE
A proposal for a new game design tool
Abstract

When creating computer games it is necessary to use extensive documentation so that everybody involved in the creative process is up to date. This paper has investigated what prospects there are to improve the GDD process to a more updated version that would be less linear in its format and more user-friendly to everybody involved.

The theory describes relevant concepts about the characteristic computer games industry and touches some points that programmers and esthetic content creators come in contact within this environment. The theory also highlights some contrasts between these professions and the game designer concerning available resources and the communication between them.

It also highlights the advantages of using graphical information over text to convey certain information faster and more ubiquitous over several channels.

This paper evaluates the game designers’ available tools which were found to be inadequate to do an effective job, mostly because they are not suited for the complex task that is contemporary GDD creation and maintenance. It was also found that there exists software technology and methodology to address some of these problems; but no evidence has been found that it has been implemented for the computer game industry in a useful manner.

Lastly, a suggestion for a framework that would use a more visual and amorphous workflow derived from the discussed theory is presented. This framework uses an open nomenclature in a node-graph structure where every node represents different entities/objects in the game and is connected to each other by their interaction between attributes.

Key words: GDD, process, graphic, interaction, design
1 Introduction

When creating a game there are many good tools, sometimes several within a single field/area. When work is divided and shuffled between different disciplines and forward in time there are also a lot of established methods and good tools to choose from as well. There is however one important transition that could work better regardless of production size, platform or genre; the transition from concept to production (Dille & Platten 2007).

It is an elusive step that has to convert abstract ideas and concepts into tangible (and often measurable) items and rules that can be understood by everyone that will work with the game (Callele et al 2004, Kanode & Haddad et al 2009). The conversion from idea to practice is a difficult one at best since it is usually the esthetically inclined designer(s) that writes down a number of ideas of how the different parts of the game should behave (among all other aspects of the design like appearance and feeling) (Adams & Rollings 2007). This text is going to be read by people that want specific (less interpretive) format of the information since they are going to create their pieces based on what is available in the documentation. This requires that the documentation is extensive and that a rigid version handling system is in place, all aspects of game production is very iterative.

The documentation is handled differently for every game since there is no established standard on how it should work; experience usually determines how and is guided by personal preferences from management (Manninen et al 2006). Regardless of how it is done this information is usually referred to as the game design document(s) or GDD. These are intended to be living documents that helps everyone involved to understand as many relevant aspects of the game as possible and stay up to date with the subsequent changes that arises, in theory.

The GDD is however almost never complete, or at least representative of the actual finished game when released. This is not crucial for games’ success but it is a powerful tool that is decreasingly used from the middle phases of production onward since the limitations of what it can provide. This limitation comes from the increased manual work linked to the increasing complexity of the game’s capabilities that take too much time to document (Kelly 2012). The aim of this paper is to provide a suggestion of how to upgrade the process from the current GDD-approach to a more integrated functionality that has more useful features for longer.
1.1 Same concept, growing mass

A game could be described as a closed interactive system where the user tries to execute specific combinations of instructions in order to achieve completion of defined tasks to progress to an end goal, there are several more descriptions (Salen & Zimmerman 2004) and attempts to give a definitive description but this one is adequate for this paper. This concept has not changed since the beginning of games.

Creating games requires some sort of method/tool where every entity in the system could be given a set of attributes and how they interact with each other, entity to entity but even attribute to attribute. Some of the attributes even have to be able to change depending on specific circumstances, keeping track of all the variables and producing a consistent and rigid model of game mechanics is very complex.

Today this is done with tools and methods developed for games that was fairly simple, the technology to do so back then was, in comparison with today’s technologies, simple but adequate. The games have evolved; the methods and tools have not seen a parallel evolution in capabilities.

This paper will investigate if it is possible to construct a tool/concept to generate a general system that can describe any interactive system and make the resulting description a framework for subsequent design decisions.

1.2 Existing tools and methods/systems

To get to a finished concept of a game is a process in itself and there are many paths to choose from, regardless of path/method taken the ideas and decisions made must now be organized so they are easy to grasp for anyone that was not part of the decision process (Adams & Rollings 2007).

There is no universal standard for how a GDD should look like, there are many templates that one can download and use, but they are only guides. In the early stages of development the GDD structure is roughly pieced together from what worked before that is still applicable and figure out how to fill in the gaps with a new workflow (Adams & Rollings 2007).

Once the core concept is decided it becomes the established reference/framework and should never change since all subsequent decisions are based on it. Game production is a very gradual and iterative process, as more details and behavior are added so are consequences of the interaction with them, the final overall behavior is impossible to predict and must repeatedly be tested to see if it will work but also if it will be entertaining enough (Schell 2008).
1.3 Current shortcomings

The software solutions available in theoretically assisting with structuring a GDD as intended in this new context are too specialized or generic to be efficient. Also, none of the solutions found could provide an adequate holistic strategy to either be integrated, or export any result created, into an established workflow. In order to create a comprehensive image of what the game is supposed to do that is easily understood by others today, a designer have to use several software packages and manually linking them together, this is a very error prone and time consuming method (Dille & Platten 2007, Kanode & Haddad 2009).

Another issue that is often changed/updated and hard to describe is the behavior of the entities in the system (Dille & Platten 2007). Behavior is complex since there are often several factors that determine what happens at specific circumstances and keeping track of all interdependencies is a massive undertaking. When changing one aspect of one entity one could unknowingly offset a finely tuned balance of an entire section.

Since it is humans that come up with everything that is to be included in a game it is impossible to think of everything in the beginning and getting the things you thought of right the first time. It is also difficult to see how any new detail will fit in the big picture; will it contribute, be a burden or introduced in the wrong place?

The GDD becomes more separated from the main workflow rather quickly as soon as production begins; when the framework is established and populated with more details the workforce rely less on the GDD because the creators become more familiar with the inner workings of what and why, updating the GDD becomes more of a burden in the end (Callelle et al 2004, Kanode & Haddad 2009, Kelly 2012).

1.4 Proposition

This paper is a preliminary study to determine if it is viable to upgrade the current GDD process with a more contemporary and up to date tool/method that is better suited for the modern computer game industry. It will focus on an interface and structure that take advantage of the capabilities of graphic and spatial navigation. So the research question is:

*Is it viable to replace the text-based descriptions of the interactions and capabilities of proxies in a GDD with an interactive node-based graphic navigation interface and maintain equal or improved functionality?*
1.5  Scope
This paper will only use digital games as examples of interactive systems since they are more complex than other systems like web-pages or kiosks, if this concept works on digital games it will work on other interactive systems as well. Note that this tool is also suitable to use in the creation of physical games as well and will have the capabilities to define them as standard.

This paper describes a tool intended to create a structure for an interactive system, not a complete set of methods and/or templates. This tool is intended to organize the creative process and save the users’ time and needles iteration. It is a foundation for later decision making when looking at amount of hours spent creating the assets.

This paper is not trying to prove that there are universal solutions in how to make games nor how to structure them or describe them; it will try and describe an approach to give the designer a different tool that is open and flexible but still structured.

1.6  Target Audience
This concept should be beneficial for anyone who is involved in the creation of user interactive systems such as games and/or web-pages.

Conveying what a designer means and wants into a more technical and practical format is notoriously difficult to capture, this tool would help both the esthetic and technical side to meet half way. The designer will have to define the interaction and behavior of the system with a set of rules (that also is defined by the designer) which has unambiguity to programmers thus showing the intention of what is to be achieved more clearly.
2 Theory

This chapter describes four major sections. First a very superficial glance on the industry as a whole and describing some of the idiosyncrasies present. Secondly, a more detailed description of the smaller units of game design which the new proposal will improve upon. Thirdly, a few important considerations to the recipients of the output that need to be taken into account are described. Fourthly, some basic concepts of human perception and graph theory which will be the basis of the new graphic layout of the new concept/tool will be addressed.

2.1 The gaming industry

For anyone not familiar with the gaming industry there are three main characteristics that define it well: diverse, iterative and changing (Adams & Rollings 2007, Dille & Platten 2007 and Schell 2008).

It is diverse in almost every aspect. How things are done, in which order and why are all different no matter where you go. The reasons for this trait will be somewhat explained throughout this paper but the effects it produces is more important to address and is one of the defining factors for the design of the tool Granode.

The iterative process is inherited from software development. Since prototyping software is very fast and cheap is can be done many times and every mistake that is made has a much smaller potential to have serious effects. The hedonic and irrational nature of human emotions and taste for entertainment makes this method particularly effective in figuring out what is “fun” to play (Salen & Zimmerman 2004, Schell 2008).

Everything in this industry is changing, often fast. The technology, the complexity, the interactions and multi-channel distribution et cetera are constantly changing and evolving. This is what drives the diversity of the industry somewhat and vice versa (Kanode & Haddad 2009); it is another important aspect to take in consideration when trying to comprehend how the companies make decisions in this environment.

This is the background of an industry which has to produce consistent products that must be well received by the consumers immediately or there might not be a second chance to make another game. Having a good grasp on what is going on in the production is essential, a process that start with the GDD.
2.1.1 Production structure

When describing the theory of how a game is made this paper identifies three levels of abstraction: theoretical, planned and actual, these moves from theoretical towards what actually happen. On a theoretical level, game makers agree on how games should (and are) made, they all use a same basic principal model (fig 1):

![Basic production model](image)

Fig: 1 Basic production model

Also when it comes to what roles there are and what they do is consistent throughout the industry. There are minor tweaks on responsibility issues between companies and actual work tasks but overall comprehension within the industry is clear (Adams & Rollings 2007, Manninen et al 2006).

When a game production is planned the basic model is refined to a large extent and the sometimes vast differences between developers become evident. What model suits the production best depends on factors like company size, age, nationality, corporate culture, target audience et cetera. Even within the same company there could be different production models because the end products are so different (Manninen et al 2006).

When actual production starts all the real life situations comes into play that will modify any plan ever further. Before any solution is tested they are theoretical, nobody knows if it will work until the results are in. This comes in all areas like software comparability or dividing teams between floors (Dille & Platten 2007).

Contemporary technology moves so fast that it is common for a sequel to a title to have substantial improvements in capabilities, features and design. This also applies to the technology that develops the games which means that tools become more streamlined and more capable, including the workflow they are a part of (Dille & Platten 2007). Seasoned companies know what to keep from earlier projects and what will have to change.
It should be said that contemporary companies that has been active long enough to release a handful of games or more have learned enough from their own and others previous mistakes to have (at least internally) a much more stable environment today than six-seven years ago. The industry itself has matured enough since the turn of the century to have serious business models and is well integrated in today’s business culture (Wikipedia - Game Companies 2012).

Today’s interconnected world allows smaller companies to come in contact with their target audience and other developers a lot easier than before the modern Internet culture and thereby learn faster than their predecessors (Gamasutra.com 2012).

2.1.2 Posts in production

When computers came about the computer games soon followed, up to the mid-eighties one game was usually made by no more than a handful of people and when technology improved so did the demand for better/bigger/faster games, today it is not uncommon for a big game having 200-300 people working on it (Dille & Platten 2007). What a person does in a company is among the few things that have become a standard in the industry, mostly because it is based on skills from mastering the available tools.

Here are the common jobs titles divided into two groups that are likely to have direct or no direct interest in improving the GDD process:

Direct interest:

- Art / Animation
- Audio/Music/Sound
- Executive / Management
- Game designer / Level designer / Creative director
- Producer / Director / Project Manager
- Programmer / Engineer
- Writer / Scriptwriter

No direct interest

- Accounting / Finance
- Business Development / Analyst / Legal
- Customer Service / Tech support
- Educator / Instructor

Continued…
- Marketing / PR / Communications
- Network / System admin / IT
- Quality assurance / Tester
- Sales

It should be said that all of these areas of responsibility exist in every game production; it is just a matter of dividing it up among the employees (Adams & Rollings 2007). In smaller companies there is a more crude/generic division of labor but the ratio between spent man-hours are basically the same regardless of size; administration spend about 10 %, the creative arts about 40 %, management 10 % and programming / technical uses about 40 %. These fluctuate somewhat during production but are a typical average (Gamasutra.com/jobs 2012).

The point is that every one that creates content and those who supervise the production would benefit mostly from an improved GDD process (posts within the blue rectangle in fig 2 (exception QA/tester)). The company as a whole (and every one working there) would eventually benefit from a more streamlined production in the long run but that is theoretical and will have to be proven at a later date.

Fig 2  A rough approximation of posts interaction within a production
Within the production rectangle, the direct usefulness of the improvements Granode will offer decreases somewhat the further up from the “Game Designer” post one goes in the diagram. Any post outside the production rectangle will only benefit indirectly.

### 2.1.3 The iteration process

There is no formula to create a successful game and there never will be. Words like “game”, “fun”, “play” and “entertaining” are notoriously hard to agree upon since it is ultimately based on experience and preference of the individual (Salen & Zimmerman 2004). There can be an agreed technical definition of what any of these words mean (they all exist in a dictionary) but not what they represent for the individual.

Even the word “successful” needs personal definition; is it the game that made the most money or has largest number on-line players or some other criteria? Again, it depends on who is asked.

In order to create a successful game it is necessary to iterate the design, the game play, the story and every other aspect of the game as many times as practicably possible. The iteration process is not only to find out if the game runs smoothly on a technical level but mostly if it is enjoyable to play. It is very seldom the original idea works immediately; everything needs tweaking, complementing or a complete re-design (Dille & Platten 2007, Schell 2008).

The human mind is good at processing complex information on a more subconscious level; the feeling of what it is like to play is comprised of many levels and is impossible to dissect and analyze individually and declare a hedonic value-matrix. Manny instances of fun game play have come from unexpected constellations that never would have come from strict addition of “fun components” of various games (Dille & Platten 2007).

This process is hard to make time for since it takes time and/or resources from production. Experienced managers make sure there is as much time to playtest as possible; this is not to be confused with beta testing which comes later during the later process of production. Play testing is done in the early part of production onward to see if the design team likes what they have created, beta testing is (depending on production) either exhaustive stress testing of less probable scenarios and/or play testing with gamers not involved with the production (but interested in the outcome) to give their verdicts if the game is on the right track (Adams & Rollings 2007, Dille & Platten 2007).
2.2 Game design documents (GDD)

The game design documents is the collection of descriptions the game designer(s) has put together to explain the entire game, every aspect is covered. There are several sub categories within the documentation, for smaller productions it could be one document with different chapters while bigger productions have separate files since they become too cumbersome to handle if merged into one (Adams & Rollings 2007).

2.2.1 Common categories are (in a non-inclusive list):

**Technical**

This handles hardware/software requirements and technology comparability, i.e. should the game be available on PC-gaming computers or only on consoles et cetera. This section also covers requirements and specifications for functions that will affect third party systems, functions like multi player or integration with social media platforms et cetera.

**Characters**

Here are descriptions of the different entities a player can encounter; it could be anything from a simple rat (to add ambiance) or an advanced AI-driven character that guide the player through a haunted house. The level of description per character depends very much on what its purpose is. A major character could have a several page long background story complete with psychological profile and numerical stats of all capabilities. A menial drone/servant that has the single purpose of being cannon fodder for the player has a very short description with a handful of stats.

**Game play**

This section could focus on how the different modes of play should work and what the player experience should be. This is a very abstract section with few measurable criteria; description of moods, emotions and skill difficulty et cetera. The most tangible information here would be the button mappings; press “X” to jump and “Y” to run et cetera.

**World/Story**

This section describes the game world in general. If it is on another planet or in a fantasy world there are descriptions that explain why things are the way they are so the production team can relate to the abnormal circumstances with better understanding. If it is our world, this information could be more focused on local/personal history to shed light on why this game takes place where it is (Dille & Platten 2007).
2.2.2 **Ubiquitous illustrations**
Regardless of what the player sees on screen, it needs to be designed, built and integrated in the game world. The majority of the design is done through concept designers who sketch the most important visible elements that exist in the game (characters, buildings, vehicles, equipment et cetera). These sketches are used as often as possible to illustrate what something looks like since it removes ambiguity about appearance and provide detailed information much more efficiently than text (fig 3). Sketches are also preferred over text since it sets an atmosphere that helps the team “getting in the mood” (Adams & Rollings 2007).

If the entity is something that moves it also needs to be animated and/or have a behavior assigned to it. This is harder to illustrate but can be done with sequences of images and explanatory text that describes what happens.

![A concept sketch](image)

**Fig 3** A concept sketch

2.2.3 **Definition aspects**
The categories mentioned above are mere different aspects of looking at parts of the whole. The GDD could be divided into any number of ways depending on how the production is structured; as long as it is possible to create the game (as intended) from the information in the GDD alone it is adequate regardless of structure (Schell 2008).

2.2.4 **GDD Format**
Even if the technologies of creating games have moved forward at a phenomenal pace the fundamental way of creating a GDD is still with text documents (Dille & Platten 2007). The ability to share over network is
mostly to allow instant access to updates, not collaboration since only the designer(s) should be able to edit.

Even if the medium still is text (emphasized with images) the designers are no longer confined to word processors (and files); they can use different html-based solutions and publish the information through a web browser. There are still some drawbacks to using this exclusively; poorer (but getting better) spelling/grammar checks and more hands-on coding to format the layout (unless templates are used).

Even if it is easy to integrate graphics as effective information condensers there are still a lot of information that is better conveyed through text, such as intentions, reasoning, history, and many more. The biggest problem considering text formatting the GDD is that as time passes during the concept phase and into production phase the game will become either more complete or more complex, which ever it is it increases the amount of text that need to be written, maintained and eventually (hopefully) read (Callele et al 2004).

2.2.5 In reality

This quote from Tadgh Kelly, who has been working with games for twenty years as designer, writer, producer, startup founder and writer for several game related outlets, says:

“... In theory it is supposed to be a living document, which means it changes over the course of development. However a long-standing joke in the industry is that game design documents are judged by their weight (in paper) rather than their content because nobody actually reads them.

In practice game design documents are often weighy tomes of promises, stories, executive summaries, untested gameplay ideas, business plans and all sorts of other redundant details. They are usually badly written, poorly edited, bizarrely structured and repetitive. They also regularly become the subject of contentious disputes between developers and publishers because each interprets their meaning differently.”

- Tadgh Kelly - What Games Are, 2012

A major reason for this collection of defective traits is that as the number of words needed to give adequate (and later added) descriptions increase, the amount of man-hours stays the same. The volume of work cannot be made in the time required (Adams & Rollings 2007, Callele et al 2004).
2.3 Entity features and capabilities

When describing what the different parts/objects of the game are supposed to be capable of they need a description that can be interpreted by the rules of the game. The rules are a set of conditions that define if something is allowed or not; if humans are to understand them easily they are written in spoken language text to describe how, where, when and why something happens. When writing code for a digital game the rules are less obviously defined to the user/player, even if there are more of them. When humans understand a rule it is interpreted from text and a larger understanding of what is logical, probable and reasonable, it can be expressed less exact (Weinschenk 2011).

A computer needs specific numerical values since it does not have the capability to understand rules, it measures values and execute instructions that are valid when certain conditions apply. This means that all aspects/attributes of every part/entity of a game need to have a numerical value that describe how much of it there is. It is also necessary to describe how an entity affects other entities and under what circumstances.

2.3.1 Entities

Since all objects in a game may not have a physical manifestation and/or representation and some concepts are described as objects (but are not) the term “entity” is chosen (in this paper) to capture all of these with an all-inclusive name. A common trait for all entities is being phased in and out of the production depending on what is being tested. Some are scrapped completely others are merged and some are re-located to a different part/level of the game (Adams & Rollings 2007).

2.3.2 Attributes

In the beginning, entities’ attributes could come and go since it is not known if they will fit in the final design or not, after a while the concept solidifies and the description is fixed. It is still possible to change their values and possible interactions until the very end since they are easy to change in themselves and are the most common aspect that changes in the final tweaking when balancing the game, keeping track of the consequences and evaluating if they are desirable is another matter, a good rule of thumb:

> For every additional attribute of an entity/asset and/or possible interaction with other entities/assets will result in additional edits for each attribute, interaction and iteration to maintain/achieve a balanced game.
In addition: The further the consequences reach of changing any of these values/interactions will result in more edits.

2.3.3 Levels
Since players do not want (or are capable) to play the entire game in one go it is sectioned up into smaller pieces called levels or maps (depending on the game), each level could exist for many reasons but the primary function with this division is to pace the game and provide variation.

Each level/map offers a variation of game play; some are designed as sneaky levels while others could be puzzle/mystery or plain mayhem and countless others with variations. Other aspects are to give visual variation: desert levels, mountain levels, urban levels; all designed either as an excuse to use a feature in the game engine or a game play idea or to drive the story forward (Dille & Platten 2007).

Since the theme of any level requires some unique elements this increases the amount of assets to be defined, made, placed and balanced. These entities must also be able to interact with some of the more common entities that are transported from other levels and vice versa.

2.3.4 Answering the “what” questions
All the attributes and interactions are supposed to answer what are the characters, vehicles, opponents and weapons et cetera capable of? They should explain both to the rules and anyone who is reading the description, different parts are relevant to different posts (2.1.2);

- Sound wants to know how many sounds to create and what they should be
- Programming wants to know what interactions with other assets there are
- Animation wants to know how many movements there are and what they look like
- Modelers want to know how/if the appearance will change during damage

Et cetera (Adams & Rollings 2007, Schell 2008)
A complex but unavoidable fact when introducing an object is that the possible influence to all other relevant objects need to be mapped. This is in addition to just define the entity in the metrics that is present in the game world.
**Example of the lonely crossbow:**

**Object related metrics:**
- Weight: ________________ 13 Kg
- Range: ________________ 150 Meters
- Ammunition capacity: ________________ 1 arrow per reload
- Reload time: ________________ 5 seconds
- Damage: ________________ 15 points
- Reliability: ________________ 98 %

**Object related connections:**
- Damage target types: ________________ See Table 1
- Exist as an object in game world? ________________ Yes
- Carries ammunition? ________________ Yes: 1
- Available upgrades? ________________ See chart X

**Connections (Player):**
- Does it need specific skill to operate? ________________ Yes
- Will it count toward the player’s inventory? ________________ Yes, 5 slots
- Will it require specific ammunition? ________________ Yes

**Connections (Rules):**
- Can fire through vegetation? ________________ Yes
- Damage on walls? ________________ No, stops arrow
- Affected by magic? ________________ See chart Y

*And so on…*

**Table 1:** Example of modifiers for a crossbow

<table>
<thead>
<tr>
<th>Target</th>
<th>Damage modifier</th>
<th>Fire Arrows</th>
<th>Armor Arrows</th>
<th>et cetera</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Humans</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Farmer</td>
<td>+5</td>
<td>+10</td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>Soldier</td>
<td>-5</td>
<td>-3</td>
<td>+5</td>
<td></td>
</tr>
<tr>
<td>Thief</td>
<td>0</td>
<td>+5</td>
<td>+3</td>
<td></td>
</tr>
<tr>
<td><strong>Undead</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zombie</td>
<td>-10</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Skeleton</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Mummy</td>
<td>-5</td>
<td>+15</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

*et cetera.*
This is an incomplete description; there are many more considerations to take into account when figuring out what a crossbow (or any other entity) can do, what that would be depends on the game it is in and what it needs to do/be. The entity should only be described with affirmative features (with exceptions if present) which take less space than explaining what it is not. This is also just an example of how this definition could be visualized and organized. However the information is presented it must be understood (ideally/theoretically) what a crossbow can do by both programmers and the esthetic creators without any other information.
2.4 Esthetic content creation

Everything a gamer experience with the senses in a game is produced by what this paper collectively calls “esthetic content” creators; this includes all the visuals on the screen, all the sounds and to a large extent the writers. There is normally a much more recognized division between the different disciplines (see 2.1.2) within this group when compared to programmers but for the purposes of this paper they can be considered as one. (Adams & Rollings 2007).

2.4.1 A pure content creation environment

Content creators want powerful and efficient tools that are easy to use. According to the interviews these tools exits today, they are also familiar, they have all the functionality that is needed and there is an established workflow to utilize them. There will always be room for improvement and increased capabilities but there are no obstacles to produce exactly what is wanted within reasonable time. The same reasons why programmers do not want “universal” tools/solutions (see 2.5.4) apply to this group.

The point being that it is not wise to integrate any tool across realms since no one on either side will appreciate any worthwhile benefits.

2.4.2 Answering the “why” questions

Content creators have a real need to know in what context their work will be placed in to do a good job. Everything should be approved by the designer but he/she does not have to design everything; that would be unrealistic, the esthetic content creators therefore have the freedom to design themselves as long as it fits with the context/style of the game (Dille & Platten 2007, Schell 2008).

Since the documentation cannot possibly describe everything in detail in the game a larger metaphorical picture to draw initial inspiration and conclusions from is provided (Dille & Platten 2007). A background story gives ample context to know what the game will be about and get the team in the right mind-set, some examples:

History

Knowing where large things (wars, corporations, civilizations, technologies, disasters et cetera) come from and how long they have been around gives perspective and provides a large time scale.
Background
Knowing where characters come from and what they have done/do gives them depth and some understanding of their personality.

Physics
If there exist some kind of phenomenon like magic, warp drive, telepathy, anti-gravity et cetera and how long it has been around gives good indications on how the world would look like.

These are just three examples of explaining some of the how and why that goes on in a game world. Theoretically the GDD do not need to explain any of these things in more detail than will be noticed in the game, but as mentioned earlier in the iteration process (2.1.4) it is impossible to know beforehand what will make the cut and what will remain. So why is it still a good thing to just produce as much background as possible?

If the initial design is somewhat loose and collaboration is embraced that design still sets the mood/style but the end product could become much more vivid and multi-faceted than if one person had made it all. This could inspire the creators, also known as the participatory design method (Grudin 2000), as long as the design is approved by one or few individuals to avoid “design by committee”. This could however also create a problem with organizing the material and monitoring that people still do what they are supposed to do.

If the game designer have a detailed and/or specific view of what the game should look like he/she need to start making those descriptions long before the collaborative work begins since one person can only make so much at a time.
2.5 Programming

Games today are not sold in great numbers because they have efficient code or elegant algorithms; games sell because they are enjoyable to play. Many attribute this to either impressive graphics and/or emotional stories. But none of that emotional eye-candy could exist as a game without the programmers. Even if the individual programmers are specialized within a certain type of programming, being a programmer is still considered a generalized and somewhat unappreciated post (Cooper 2004). But to be fair, programmers as a group have much more in common in how they think and solve problems compared to the other groups in the production team (Cooper 2004).

In order for a programmer to give as good feedback as possible on how complex certain features are and how to fit them into the main structure efficiently there are some basic concepts that must be taken into consideration when presenting a solution/concept, these apply to any programming language.

2.5.1 Traceability

All programs are collections of instructions that are supposed to do specific tasks when specific circumstances align. All instructions exist to transform specific data (or meta-data) from one form/value through a process to shape a desired result. The instructions are absolute and have no ambiguity about them, if an instruction cannot execute there is an error and the process stops (Wikipedia - programing 2012).

All values are either static (as an influential part of a mathematical function) or a variable (result from an equations). More complex functions rely on the result from other functions results (variables) and it is important to clearly see these connections and be able to trace where the information originally comes from. This way of thinking applies to any scale in programming, knowing where the data comes from and where it is going is essential (Wikipedia – traceability 2012).

2.5.2 The big picture

When organizing the general structure of any size of code it is important to know where it will fit into the larger picture. Knowing what kind of data will be processed, the amount of data transfers, what kind of results is desirable et cetera is significant information to make efficient code (Cooper 2004).

When defining properties, behavior and rules for the game and its individual parts it is important they all use the same metrics and scale. If changing the
number of properties or type of ability for a specific object after its implementation in the system the structure should support that type of change. That can only happen efficiently if the programmer is aware of the big picture and can prepare for that type of change (Cooper 2007). This is also closely related to the traceability issue covered in 2.5.1.

Being aware of the big picture and how it works as a whole is always helpful, especially when programmers enter the project in the middle of production and quickly needs to familiarize with the structure.

2.5.3 Hierarchy and tagging

Whenever a section of code can be compartmentalized and given a name, the name needs to be unique so there will not be any confusion. The section can also be copied and be slightly modified to create a similar function. It is necessary to separate the two names but since the codes are similar in function they can also be given a group/family name.

To manage this it is possible to create a hierarchy where the original code is a parent of the subsequent copies which are grouped as a sub-class under the parent. This can be done on any level as long as a consistent standard is maintained and could create a significantly large tree. It is however possible that even if two sections of code are similar and are located next to each other in this tree structure they are not used in the same part of the game, this creates an issue about how the code should be organized.

A simple way of solving this is to tag each section with key-words that are searchable, called meta-tags (which there are a few different variants of). These key words will make them easier to find in a large structure but will also allow users to organize the filtered data in categories (see also 2.6.4).

The drawback with tagging is that there is a balance on how to use them:

- Use a standard terminology that is consistent throughout the entire project
- Use enough tags per section that make it unique but not more.
- Make it possible to add tags to the library but only out of real need, not potential need.
- Make sure there is a filter and sorting tool that is adequate to the amount of tags available.

(Wikipedia – meta-tag 2012)

These two methods combined should give rigidity and flexibility to organize and find information easily, both when creating code but also when traceability (2.5.1) is needed.
2.5.4 Pure programming environment

Even if programming generally is not considered an art form the programmers are at least designers and therefore creative people. And as many creative persons; programmers want tools that are easy to use, efficient and as powerful as possible, programmers have such tools today.

Programmers like their tools because they are familiar, they have all the functionality that is needed and there is an established workflow to utilize them. It is not uncommon that skilled programmers create their own tools if the standard kit has any shortcomings (Cooper 2004).

If a tool is supposed to work across disciplines it should not be expected that it will perform well in any of the disciplines it is used, also known as the Swiss-army knife syndrome (Urbandictionary.com 2012), it can do many things but not very well. It should therefore be avoided to use “universal tools” if efficiency is desired since they are not the best they can be (programmers like efficiency) (Cooper 2004).

2.5.5 IT is not programming

IT is the day-to-day tasks of management/maintenance of the physical hardware and administrative software that resides within the production.

There is a group of software solutions that is called “Production/Digital Asset Management Software” which keep track of all models, sounds et cetera that are the in-game assets, complete with version handling, back-up solutions and other tools (Wikipedia.org – DAM 2012). These assets are virtual files that are accessed through the internal network to and from the main storage, both of which IT usually have responsibility for. But these are expensive so only large productions can afford to use them, and the lack of industry standards mean that local/propriety solutions are used instead, if at all.

IT is usually handled by one or more programmers (or network technicians) but they do not necessarily work with the game itself, they do communicate with people that create assets and often have opinions on how other network users should manage IT-related aspects of their work in order for the system to work efficiently (Cooper 2004).
Perceptions on a screen

There is a lot to say about how humans perceive the world, partly because it is a very complex subject but also because we do not fully understand it yet and therefore have competing ideas, beliefs and models to describe various aspects. This paper will therefore only shortly cover some of the most relevant basics within this topic that has been adopted as useful enough to be used frequently within the field of interaction design.

The theories covered here are chosen that are relevant to a tool that is primarily a visual interface. The intention is that the tool will be flexible and open enough for substantial specialization and easy expansion.

“Design for the real world!”
– Don Norman, Talk at Business of Software 2009

The Don Norman & Jacob Nielsen goals

Don Norman and Jacob Nielsen are two accomplished writers and lecturers within the field of interaction design and started the Nielsen Norman group in 1998 (Norman Nielsen Group 1998) and are considered pioneers of user friendliness. They have many guidelines on their site on how to create and design interfaces, digital and physical, but there are design principles from Mr. Norman and usability heuristics from Mr. Nielsen that is at the core of it all. Some of them overlap in meaning so they are presented and paraphrased in the same list:

Visibility
If it is important and/or used often it should be highly visible and easily accessible. If it is less used and/or important it should be less visible but still easy to find.

Feedback
Show the user what is happening, show progress and/or change if there is any. Do not delay with responsiveness or information.

Restriction
Do not give a user the opportunity to make an unrecoverable mistake; if necessary, explain why the restriction is in place.

Freedom
Be as accommodating as possible with user exploration; make it easy to correct any mistake made. Make user advancements in navigation and execution possible, (enable shortcuts/hot-keys and other fast commands).
Consistent
Use consistent language, iconography, design and methods et cetera at all times.

Affordance
Give an objects’ function a supporting design to the extent that it is evident and practical.

Mapping
Make it as clear as possible what effect any control has.

Recognition
Be as familiar to the user as possible, relate to their world with terminology they understand. Recollection is better than memory when repeated tasks are needed.

Help
If the design still fails to enlighten the user or external factors come into play, help should always be easily available.

These are desirable goals and the real world might not allow all of them to be present at the same time, especially within a limited timeframe and budget. There will probably be some sort of compromise and the specific situation determines what needs to be prioritized. This balancing is sometimes referred to as information architecture which is a specialized skill set that categorizes of information into a coherent structure, preferably one that the intended audience can understand with no ambiguity and then easily retrieve the information for which they are searching (Resmini 2011).

Complex is not complicated if explained right
If something is complex it cannot be reduced and still be representative for what it really is, it has become a simplification. Complexity is not necessarily a bad thing, complicated is needless complexity. If something needs to be complex to work it cannot be simplified beyond a certain point. It is not suggested that all cognitive loads should be removed, just reduced or structured until it is as simple as possible to use. This practice is sometimes referred to as the purpose/goal of information design (Wikipedia.org – Information design 2012).
2.6.2 Diagrammatic visualization

Diagrammatic philosophy is the study of the understanding of concepts and ideas visualized with the use of diagrams and imagery instead of by linguistic or algebraic means. It is a sub-set to information visualization which presumes that visual representations and interaction techniques take advantage of the human eye’s broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once (Thomas & Cook 2005).

2.6.3 Text versus Graphic information

Even though text (language) is extremely versatile in describing anything it is still a linear flow that requires the reader to either start at the beginning and read until the information ends or the reader has had enough or that the reader can understand the context of the information well enough making it possible to start anywhere and progress forward from there. Regardless where the reader starts, reading is slow compared with viewing graphic information and must be consumed in a specific order to maintain its meaning (Ware 2004).

The debate over text versus icons is somewhat decontextualized. Text has the advantage of being less ambiguous if proper wording is used, but it becomes less universal in international aspect. Icons are easier to recognize but harder to remember meaning of. The best solution is icons with explanatory text (Norman 2009).

2.6.4 Information sorting

The approach of diagrammatic method has two main benefits when creating complex systems. First, when creating the system piece by piece it quickly becomes apparent that it does not take many connections to make it complex, this could hopefully give a more thought through system that is more effective than an improvised/ad-hoc structure. Second, when working visually with shapes and patterns it becomes easier to navigate and associate information since the human mind is better at recognizing patterns than remembering information alone (Ware 2004), memory also lasts longer when they can are associated in a context (Weinschenk 2011) which the diagrammatic GUI provides.

Filtering

If there is a large amount and/or many different types and/or high density of information in one place it needs to be filtered by a competent filter system
that is editable. The expression “Information overload is a filter failure” (Shirky 2009) captures this issue elegantly.

**Exclusion/inclusion and sorting**
Choosing what to take into consideration and what not is a simple and effective way of reducing unwanted information. However, it is not a clear division between what is desired and not desired as fig 4 shows.

![Fig 4](image)
The four states of knowledge

The goal is to place all available information on the “known” side of this graph. Until the user is satisfied the different pieces of information will move from the right side to left. The information that is already known could move up or down depending on what comes from the right. The grey area is where information that has not been evaluated lies until its relationship to what is known becomes clear.

This process is a mere categorization of importance; it also needs to be sorted by priority. All information is not valued equal and can be listed differently by applying diverse criteria; this is useful when doing comparative analysis (see 2.6.5).

**Structuring**
The most important aspect of finding entries organized by others in a non-linear structure (such as information on a web page or the isles in a supermarket or functions on a remote control) is that it follows some sort of logic that the user can relate to. Clear and descriptive naming helps but it is the presence of a structure with clear categories that help our human brain to
navigate the problem space most effectively; any option available to let the user organize the information by him/her gives a feeling of empowerment and control (Weinscenk 2011).

2.6.5 Comparative analysis

When comparing abstract concepts such as price or density or any other value that can be measured it has to be done with numbers or something with equal representative magnitude (Ware 2004). In a computer game, even attributes like weight and distance could quickly become abstract since it is not necessary to use realistic amounts. But they do need to be weighted relative to each other with the same scale to be predictable in the simulated environment. Everything is measured in numbers, the indicators used to measure and to adjust use numbers. A balanced game makes sure the supply of what is desirable is hard to get but obtainable enough to keep going; this is one of the most important aspects of making the experience challenging and entertaining (Schell 2008).

All games are disguised collections of equations of addition and subtraction where the outcome is uncertain and therefore adds to the excitement (Schell 2008). The psychology of why that is and how to achieve positive emotions from a game is an enormously complex issue and lies well beyond the scope of this paper. The games that rely more on the players’ usage of the actual numbers are for example strategy and role playing games while those who rely on that the numbers are well hidden are for example casual games and hand/eye coordination games (Adams & Rollings 2007).

Weighting abilities and capabilities

All numbers that we use in text represents or symbolize amount, the only exception is if the number is used in (or as) a name. Our true understanding of this concept works pretty well for notions that are directly indicated like weight, distance and quantity; a higher number is bigger, longer and more of X (Ware 2004).

When comparing numbers it is much easier to understand the difference if they are compared side by side and with relative/representative graphics, see fig 5. It is also helpful to provide (other optional) numerical references and/or using one of the values as a base-line to compare the rest with, here the biggest (380, red) is the base line for the others (Lidwell 2010, Ware 2004).
There are several more ways of helping the reader to put things in perspective, again it depends on what it being compared and what the reader wants to know.

This type of comparisons is something that is done on everything that has any influence on any aspect in a game to see if that/any entity is balanced with its surroundings. The point is that when creating a game there are a very large number of comparisons and calculations that need to be made to avoid any imbalance (see 2.3). It is also impossible to know what kind of analysis will be needed for any specific game (see 2.1.3). Another example of useful comparison is of relative sizes of in-game characters (Fig 6).
2.7 The Network grid

Any process, natural or artificial can be visualized, often multiple ways; the reason is to make it understandable since it is too complex or too cumbersome to understand well enough by description only. When dealing with a naturally occurring phenomenon the division into categories and units is artificial in order to create a simpler, understandable system. The division can be made in a vast number of ways depending on what the visualization is supposed to show.

Artificial systems are per definition already defined because they use categories and units that describe that structure. “Artificial” in this case refers to completely theoretical concepts that describe either virtual/digital processes or constructed models of real world processes. Example: a water treatment plant is not artificial, the theoretical model to describe it is.

2.7.1 Large scale specifics

At its basic level there are three main components to a visual representation model whether it is a graph or flowchart:

- The nodes that carry the information
- The relationships that is the connections
- The operation that the connections preforms

(Wikipedia.org – Graph-theory 2012)

At the next level it becomes a lot more complicated. Depending on what the system is supposed to make clear the nodes/information, type of relationships and type of operations become specific because the representation is to understand why/how/when the process provided a specific type of result.

The variants of visualized systems exist as flowcharts and every form of graphs (except for mathematical function graph, including diagrams, which is different). Flowcharts have direction(s) (flow) while the other graphs have connections of influence and/or commonality of some sort (see 2.7.3).

All representations also need limitations on how much it shows since every node can be connected with some sort of association to another. A good example of limitation is looking for information on Wikipedia; the relevance/interest on the subject will be enough on some links to follow but not enough on others.

All this shows that the broad-spectrum tool of graphs cannot be universal except on a very abstract and conceptual level.
2.7.2 Definitions
Ontologies are the structural frameworks for organizing information and are used in several areas as a form of knowledge representation about the world or some part of it. Particular meaning of terms applied to a domain is provided by the domain ontology (Wikipedia.org - Ontology 2012).
Since domain ontologies represent concepts in very specific and often varied ways, they are often incompatible with each other. Different ontologies in the same domain can arise, when merging two before separate ontologies, due to different perceptions of the domain. It could be based on cultural background, education, ideology, or because different representation languages were chosen (Wikipedia.org - Ontology 2012).

Specification language
With ontologies in place the system needs a specification language which is a formal language used in computer sciences. Unlike most programming languages that are directly executable, specification languages are used during systems analysis, requirements analysis and systems design. They describe the system at a much higher level than a programming language. It is considered an error if a requirement specification is cluttered with unnecessary implementation details, the specification is meant to describe the what, not the how (Wikipedia.org – Specification language 2012).

UML
Universal Modeling Language is a standardized general-purpose modeling language in the field of object-oriented software engineering. It is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development (Wikipedia.org – UML 2012).

2.7.3 Flowcharts, spreadsheets and graph theory
The three methods that exist today that show a graphical representation of systems all have their strengths and shortcomings:

Flowcharts do not usually process data but shows steps and paths (algorithms) in a process. The structure in a flowchart is static because it is supposed to show a specific process and what result one can expect when specific situations are enabled. The actual flow of data is shown in a data flow diagram which does not concern with time/steps but more with type of
data and the path between start and end (storage) points (Wikipedia.org – Flowchart 2012).

**Spreadsheets**
A spreadsheet is a grid of lines and columns that intersect in cells. The lines describe one data type and the columns are another. The spreadsheet’s major strength comes from the capability of linking multiple sets of cells to form complex calculations.

Spreadsheets can be used to calculate numeric in a complex system and it is not graphic, the data trends can be plotted as graphs in 2D (sometimes in 3D) to show the size and change of the data as a function but not the structure of the system (Wikipedia.org – Spreadsheet 2012).

**Graph theory**
Graph theory is the science/study of mathematical structures used to model pairwise relations between objects from a specific collection. A graph consists of a collection of vertices/nodes that are connected with edges, usually called node-link diagram (fig 7).

![Node-link diagram](fig7.png)

Fig 7          Node-link diagram

Graph drawing is an area of mathematics and computer science combining methods from geometric graph theory and information visualization to derive two-dimensional depictions of graphs (Wikipedia.org – Graph-theory 2012).

There is a vast amount of material on the subject of graph theory where seemingly small subtleties can create a whole new sub-field of research. It is all very theoretical and math-oriented and does not apply to the practical criteria of the functions of the final tool. However, the science involved will
have a key part in the development of the coding side of the information architecture. There is a more practical application of this theoretical field called graph database.

2.7.4 Databases

A database is a collection of data that is structured to suit a retrieval model that is shaped by the functions required by the system it supports (Wikipedia.org - Data base 2012). The actual retrieval of the data is done by a database management system (DBMS), like in graph theory (2.7.3) there are many different variants (of databases, their management systems and how to build both of these) that has large amount of information about differences in functions and purposes. For the purpose of this paper, only a short description of the most interesting variants will be supplied here:

**Active database** includes an event-driven architecture which can respond to conditions both inside and outside the database.

**Graph database** uses graph structures with nodes, edges, and properties to represent and store data. This is any storage system that provides index-free adjacency, every element contains a direct pointer to its adjacent element and no index lookups are necessary.

**Object-relational database** provides a middle ground of sorts between relational databases and object-oriented databases. The object-class data resides in the database and is manipulated collectively with queries in a query language.

When creating the data in a relational database management system (RDBMS) a user creates lists of items and assigns them attributes with a value.

Example: a phone book (the list) where every person is an item, every contact method is an attribute and every phone number or email address is the value. These attributes can also be expanded into several sub categories through a separate list which can be connected with an indicator/pointer. These lists are completely passive and serve only as data repository for retrieval through queries (Wikipedia.org - Data base 2012).
3 Method

This chapter will give details of the research methodology and the reasoning behind it, covering data collection, interview selection, execution and method for analysis.

3.1 Scientific premise

The information about contemporary workflows in this paper comes from literature research and qualitative interviews which is the base for most of the claims presented. The theories about mental and cognitive models come solely from literary studies.

To determine if the contemporary GDD workflow can be improved without building a prototype in this early step a deductive reasoning strategy was employed to see what logic and/or patterns are present in the claims found in the real world (interviews) and literature.

It is possible, given the large variation within the confines of this subject, that something being true in one case is not necessarily (completely) true in another. This makes some statements more inductive and probable assumptions. Whenever a broad situation is described it is considered a generalization of specifics that could vary somewhat and make any conclusion somewhat less accurate but still within reason.

3.2 Collecting data

To have a basis for reasoning and a structure to form a hypothesis it was deemed necessary to build a solid foundation in established literature and interview people working in the industry.

In addition to the literature, qualitative interviews were chosen because there is a lot of context connected to the facts and since all workplaces are different in various aspects it is not prudent to assume anything that could be asked, it was deduced that questionnaires would not provide that possibility in this case.

Interview method:

The subjects were chosen by two criteria: Availability and experience within their field, the subjects were contacted via e-mail or phone to determine a date and place for the interview. The interviewer has had extensive previous contact with all of the interviewees.

The interviews were conducted eye-to-eye on all occasions except one which was conducted over Skype. All interviews were recorded (with permission) and answers digitally transcribed and compiled afterwards.
**Literary study method:**
The literary study was conducted in two main stages and several intermediate cross-reference checking steps. The first stage was to familiarize with the available material and build an ample set of questions that could be confirmed or dismissed in the interviews. The second stage was specifically structured to find collaborative data on new information discovered in the interviews. Any new set of information was searched for in on-line sources and in previous printed literature.

**Rejected methods:**
It was deemed not practical to strictly observe a game designer in a practical manner since the kind of work he/she does is sporadic, highly mental, physically static and slow. In addition it is very secret work and the game companies are not comfortable letting people walk around asking questions. Non-disclosure agreements can be signed but it is still disruptive for the workforce to explain how a complex system works.

Questionnaires are very rarely returned since the workload is usually stretched and filling in a form that is not related to production is going to have a very low priority. It was also determined that questionnaires would not work if the industry as a whole was as diversified as indicated, the amount of questions to be planned to get a comprehensive and true image would be too great.

On-line communities are either for end users (gamers) or developers, and it was too academically cumbersome to justify and verify valid access to developers that have relevant experience.

### 3.2.1 Source selection

**Interviews:**
The people chosen for interviews were a strategic choice since they could provide a comprehensive picture of the gaming industry in Sweden from the 1990s up to today; the eight subjects finally interviewed were the ones that had available time.

**Literature:**
The printed literature chosen comes from a substantial on-going sorting process which has been active for about four years up to the writing of this paper.

Articles were chosen from an extensive search in on-line repositories such as Google scholar and similar.
3.2.2 Execution

Literary information derived on the subject of game design was mainly retrieved from books and partly published articles and partly on-line articles from game-centric forums/blogs. The interviews were conducted in two formats, person to person and over Skype. Both used the same method of few prepared questions and the rest were unstructured to maximize flexibility. The prepared questions can be found in section 3.4. The interviews lasted for about one to two and a half hours on average; one interviewee was asked questions on multiple occasions under a period of two days.

3.3 Analysis method

**Literature:**
All literature was subjected to comparative analysis to see if there was a noticeable trend in what type of issues that was brought up concerning GDD. The published articles was reviewed first and then compared to each of the books (listed in references). The interviews were done after the first complete literary analysis partly to confirm that the Swedish game industry had the same problems as the American (which is solely represented in the literature) and partly to see if any details were missing from the literature. After the interviews, the literature was reviewed again to find confirmation and/or similarities to the additional information provided. Some data regarding more intangible subjects like design thinking and design mentality were collected from recorded talks/seminars available on-line.

**Video sources:**
The videos are merely included to give context and additional insights from other perspectives, like the loosely referred blog articles.

3.4 Interview questions
Since the interviewees were selected for their different knowledge areas only question that overlapped were asked more than once. The rest of the interview contained questions that arose from the individual answers. Some probing questions about the feasibility of the concept of Granode were also asked but these were mostly conceived on the spot and were phrased differently between interviews.
4 Findings & Analysis

During the course of this study the main areas have been consistent between sources when they describe how the industry works and where it is headed. There are of course local variations and they affirm that the industry is still flexible enough to evolve and adapt to the contemporary business world and mature enough to survive long enough to be taken seriously.

4.1 Literature

There exists a very large selection of literature on the subject of making games, everything from managing interactive productions down to how to create minute details. However, the amount of new information on the practical aspect on being creative and structuring production decreases sharply after five-six books. Mainly because going beyond the theoretical level, (see chapter 2.1.1) becomes too specific to mention in any particular detail since it is likely it will not apply elsewhere but also there is little else to be said on the subject after that many books. Production-specific books can give insightful anecdotes for individual games but they seldom mention the earliest conceptual ideas with more than a couple of sketches.

A vast majority of the blog-articles did not give any useful specifics about their problems, but it provided the larger image to give context to the specifics heard from some of the interviews and warnings of pitfalls from the published literature.

4.1.1 Video sources

The three on-line sources in video format was not possible to cross-check with any other sources than a few blog articles, partly because there was no other formal source available and partly because of the nature of the content, a seminar is often about a specific subject and they seldom have external a reference on where the information comes from, sometimes the source is mentioned in the presentation. But since the speaker was renowned enough to give a seminar at such a venue in the first place they should have some gravitas.
4.2 Interviews

The interviews covered and overlapped three main areas; game production, technical issues and structure/management, these were deemed necessary to have good grasp on to get a relevant answers to evaluate the question in the proposition.

4.2.1 Overlapping interview questions

Few internal patterns present other than what regarded the GDD.

Questions (paraphrased):

- Describe what your title does and if your tasks are representative.
- Describe your interaction with the GDD (or equivalent)?
- How important is your documentation to others where you work?
- How are your tasks dependent on the GDD?
- Anyone responsible for documentation where you work(ed)?
- Can you see an alternative to a GDD?
- Where do you get information about specific details in the beginning, middle, end and after a project?
- Can you see any change in how GDD (or equivalent) has changed over the last decade?

4.2.2 Interview subjects

Names are for easier recognition and are not their real names.

Interviewee #1 (Magnus):
CEO for a small game company since 2005 that specializes in games for Facebook and has also made several commercial-driven games for big name companies such as Disney, Marvel and Paramount.

Interviewee #2 (Johannes):
An indie game maker and makes smaller casual games for digital distribution and teaches game programming half time. Has extensive experience as a programmer for game production for over 15 years and has worked for two of Sweden’s biggest game companies as programmer, including lead programmer.

Interviewee #3 (Mats):
Works as a creative consultant for a MMPORG-production (Massively Multiplayer Online Role-playing Game) that is being developed in Sweden, specialized in world creation/lore and technical specifications. Has also worked as a 3D animator/modeler in large scale film/commercial industry and has taught same for interactive media for about eight years.
Interviewee #4 (Marcus):
Works with PR and marketing with major game distributors in Sweden; also
has long experience with production of games.

Interviewee #5 & #6 (Erik & Jesper):
Works with programming for web-based interfaces for 12 and 16 years
respectively, both have substantial knowledge of databases and their
application.

Interviewee #8 (Claes):
A 1st lieutenant in the Swedish armed forces specialized in logistics and
transport. Has extensive experience with leadership, the academic and digital
structure required both from civilian and military context. Is also an avid
gamer for over 20 years.

4.2.3 Interview compilation

Game production
There are indeed a lot of different routes one can take when shaping the
initial design of a game, of the handful of productions the interviewees has
been directly involved with they gave a different story for each one. The
circumstances for each was different (time, resources, purpose et cetera) and
they did not think this will change much in a foreseeable future.
The role of the GDD is not disputed nor its necessity; all agree that it must
exist but the end purpose differ. Johannes does not write a GDD for anyone
but himself when he develops an indie game but when he worked at a larger
company he helped develop it for others to read. Magnus’ company writes it
so that the clients who order games know what is being delivered. Mats
contribute to the GDD because someone else might take over one day and
they need to know why decisions were made and why. So the definition of
what a GDD should be is not in consensus because the organization it is
created in has different criteria of what it needs. But the GDD is almost
always written for someone else than the game designer, who usually is the
one who writes it (see 2.1.3).

Even if the concept is the same as before there has been a slight shift in what
level of detail is included. Since it is a very large document to begin with it
has become too impractical to include the finest of details like values on
specific attributes on entities (see 2.3), they change too much and does not
contribute to the wider picture enough to be described exactly. The iteration
heavy pieces are placed in what is loosely described as development
documentation and are now a more informal part of the GDD that are
compiled at the end if deemed necessary. What constitutes as too iterative to include depends on the production, this is merely something that has evolved from necessity.

When asked about using a more graphic interface to view the structure of the game all with experience in the gaming industry could not see how that would help them significantly, they needed to see what could be done with such a tool before commenting on it. Those outside the gaming industry were less skeptical but could still not see a significant benefit. Basically it was too abstract for any of them to follow the reasoning at this point.

**Technical issues**
No one of the interviewees (working with games) had considered an alternative to a GDD since the issue of it being a problem in itself has not been raised. When asked if there are any alternatives to the current method no one could find any significant alternatives. When asked about the logistics of having a database structure for mapping multi-channel interactions both Jesper and Erik (who works outside the gaming industry) said that databases are common solutions to that kind of information organization. Those working in the gaming industry said that as long as the functionality was not reduced there should be no technical problems integrating a new database tool to the existing DAM-system (see 2.5.5) but the mental shift to implement something “foreign” without having a substantial track record would be too big. This resistance is more based on the preconceived notions that “what is unknown is probably insufficient”.

Jesper, Erik and Johannes said that the technology itself was mature enough to be implemented without any new concepts being developed; it just needed to be put together in a somewhat different way.

**Structure/management**
The Swedish army relies on relevant (and sometimes complex) information that is up to date to carry out their operations; the ability to handle this flow is the biggest problem with the current GDD process. After interviewing Claes it was apparent that there were limited similarities beyond this analogy on an operational scale but one aspect from the administration might be useful. There has to be complete agreement where something belong, there is no “other” category, everything has its place or can be designated a place in the structure that exists through a code and evaluation system.
When asked about possible expansion of the system Jesper pointed out that the database can be asked anything as long as there is the correct/desired type of information to be retrieved. This means that anyone can retrieve any information as long as they have access and structure the query correctly. This also means that the linear structure of a text document will be gone and that finding information will be quicker since any query will be subject based and not only categorized/chaptered.

Magnus also saw the potential to get specific updates about clusters of assets and comparing their completion with a SCRUM burn-down chart.
4.3 Situation in 2012

The term “developer” in this section refers to everyone within game production.

The term “designer” is specifically someone that designs the game that the rest is working on; he/she is either a part of a larger developers group or designs alone as an indie/hobby developer.

4.3.1 Hardware

In the interview Johannes said that the development team usually requires substantially powerful hardware because during development the code and other assets are not optimized and requires more resources until all of it can be slimmed down. Mats pointed out that designers need significantly less demanding hardware and it is very easy to upgrade if needed.

4.3.2 Software

Both within the programming and esthetic content creation software there are few obstacles that obstruct actual talent to be put forward in a timely manner. The tools that are available have been under constant development and still are. The programming tools have less apparent improvement and innovations in features since it is a very pure/minimalistic art form. The audio/visual departments have an esthetic layer that demands complex spatial displays and controls and evolves more slowly into affluent tools.

To help the esthetic asset creators, and to some extent the programmers, asset management software is usually employed to keep track of what has been made and which version of it is current, all stored in a DAM-database (2.5.5). This is very useful for larger teams because several people could work on the same asset contributing different aspects, like geometry, texture, animation and behavior et cetera The latest versions are retrieved when building the different sections (levels) of the game and are automatically replaced when an upgrade is available. Many of these solutions are local solutions that have been built to fit the specific specifications for that project, another reason for building it locally is that commercially ready alternatives are very expensive. The local solutions are also very lean and function specific if they are built as stand-alone software, if not, the solution is more of a method/procedure of using some features combined from existing software that works “well enough”.

The area with least amount evolution is the tools available for game designers. It started out as a descriptive task where using text based tools like
word processors was a natural choice. With the exception of the ability to include images with the text; word processors have not evolved much in the designers favor. With the need to explain ever more complex productions the designers started to use other software solutions to produce spreadsheets and flowcharts but that is about it. One mentionable innovation was the ability to share documentation over a network so everybody involved could get the latest information regardless of time and place, in this context “latest” means the last version written and shared, not necessarily with the newest/correct information.

4.3.3 GDD methods

In the beginning it is easy (and sometimes even enjoyable to some) to write down the descriptions of the game and its functions but it eventually turns into a chore that in the end becomes too ineffective to continue (at least as a practical part of the production).

In reality the current method works well enough to serve a purpose albeit limited. The GDD gets the process rolling and coordinates the entire workforce towards a common goal, when the details are the only unknown left the information about them is easier to come by through other channels than the inadequate GDD. This aspect is much truer in larger productions than smaller since the amount of information is bigger and less manageable.

4.3.4 Design Needs

There is no denying that many games are getting finalized enough to be sold, so the general theory of how to make games obviously work well enough for the concept to survive. There is apparently no need for a formalized universal standard in how games should be made (2.1.1); the lack of standards could very well be the driving force of innovation and diversity in the industry. This organic growth gives needed flexibility and maneuvering room to experiment with local solutions but also room for failure.

The most prominent reason for failing in game production is the lack of understanding and/or resources to have an organized design and/or production process, which start with the GDD.

As stated earlier, the GDD is necessary in the beginning but becomes obsolete and a burden somewhere in the middle of production. According to many the documentation is a joke, not because it is useless but because it becomes useless with time. A better method is clearly needed.
4.3.5 Future needs

When it comes to the graphic department there is always new features with every new release; better workflow, better realism, better performance et cetera (2.4.1). In larger productions the in-house tools made to build the game becomes more and more powerful, tools for building levels and assets intuitively is considered a necessity for being efficient. Every year there are improvements in these areas and the software developers listen to the users for what to improve next. Game designers have no such service, which still use word processors. This fact does not proclaim a need; few designers complain on the lack of tools, more focus is placed on complaining on the process.
4.4 A proposal of new method

The current method of creating a GDD is a non-linear editing of a linear medium which is one (or several) text document(s). As stated before (2.2), some descriptions are very well suited for text because meaning, depth and intentions are efficiently conveyed with text. However some descriptions are less optimal to describe with text and that is the numerical values of the separate capabilities of the entities in the game and their interrelationship with other entities (see 2.6.2, 2.6.4-2.6.5, 2.7.3). It was also revealed in the interviews (4.2.3) that this type of descriptions is sometimes intentionally left out of the main GDD because they are so unpredictable and change constantly.

This new method implies that the numerical and relational descriptions between entities be visualized with graphical methods because of two main reasons:

- That kind of technical information conveys better to humans as graphics than abstract numerals (see 2.6.2 - 2.6.3, 2.6.5).
- It is easier to organize, find and comprehend the complex interactions with graphics than linear text (see 2.6.1, 2.6.4).

4.4.1 Original inspiration

The idea for this main concept comes from a mix of MySQL, a relational database management system (Fig 8), and the hypershader (Fig 9), an application of a hyper graph in Maya (GCC software).

The graphics

The basic framework of the hypergraph and the customization of MySQL lists give the basic functionality that is suggested for Granode. But in order so handle the multifaceted nature of multiple interactions from different kind of data types some sort of filter method is needed. Meta-tags (see 2.5.3) are simple, adaptable and powerful with the right restrictions. A hypergraph is a special kind of node-graph (see 2.7.3 - graph theory) that can have any number of connections (edges) to a single node (vertex). Maya creates the rendered appearance of materials with a tool called the hypershader which is an application of the hypergraph. Every node/vertex has a function in the form of an equation, image or formula to provide a finely tunable output material by selecting specific attributes present in each of the included nodes.
Fig 8  Relational database management connecting lists

Fig 9  Different nodes connected in the hypershader found in Maya
The rules

The universal syntax to describe how the rules for interaction should work must come from a specification language (see 2.7.2) that is very direct and descriptive. Game Maker Language (GML), a collection of scripted functions native to the small 2D game making software called “Game Maker” is interesting for closer inspection. GML has drag-and-drop functions as pre-written GML scripts (an included interpreter carries out the commands indicated by the GML code when executed). Another inspirational source is UML (see 2.7.2) but it cannot be used directly in this application since it is standardized and is too abstract.

The rules would be written and stored in a separate domain, a collection of scripts and functions that would be available to choose from a list when needed, defined by the designer him/herself in order to strive for efficiency and total understanding of what it does.

4.4.2 Similar solutions

As previously stated, none of the technologies, concepts and methodologies covered to build Granode needs to be invented. They exist in other software applications as similar functions but not to the same goal. A very short list:

**Autodesk Maya**: Uses a hypergraph with multiple edge connections to create complex material simulations

**Enterprise Architect**: Uses a node based functionality to analyze and design software from requirements gathering through the analysis stages, design models, testing and maintenance using UML (Universal Modeling Language).

**K-web.org**: an Internet project that use nodes to show the vast amount of connections between historical events, people and innovations that led to the world we live in today.

**The Brain**: Software that helps to arrange anything that the user wants organized with a node-based mind-mapping solution. The nodes can be given some attributes that can be linked and edited.

**Wikipedia**: An on-line, searchable repository of interconnected/hyperlinked knowledge on the Internet.

**Facebook**: Multi-media database interface capable of simultaneously displaying text, images, video and sound connected from multiple sources with extensive filter functionality and meta-tag usage.

**Microsoft Office** (as one example out of many): Capability of making spreadsheets, graphs and flowcharts.
4.5 A proposal of a new tool

The concept for Granode is quite simple in its general design. All objects in a game are represented by a small list of attributes and capabilities (see 2.2) condensed into a node (fig 10) in a graphical node-link diagram (see 2.7.3). The capabilities of an object/node can affect other objects attributes according to the rules that the designer defines. The effect is visualized with a line/edge between the nodes. All the nodes are placed on a featureless background and can be freely moved around to be organized in a preferred manner like the hypershader in Maya (fig 9).

Fig 10 Node describing the player avatar

4.5.1 Filters

Different types of interactions can be valid and can be either color-coded or by some other means graphically distinct. It will not take long before the amounts of interactions are too many to handle effectively without a rigid filter system (fig 11) that would enable the user to separate wanted and unwanted information (see 2.6.4).

Fig 11 early concept of the filter controls
4.5.2 Rules

The rules are built from simple logic statements that use simple arithmetic and Boolean operations. The operands that can be affected are the attribute values, the effectiveness of capabilities or other rules. Two examples:

*General game rule:* Hearing distance(Enemy)

If distance **PlayerAvatar** to **ClassEnemy** <= 10 meters

Then: Alarm = On; Guards => State 2

Until: **PlayerAvatar**, Hit points = 0

OR

Distance **PlayerAvatar** – **ClassEnemy** > 100 meters

OR

(Whatever the game designer wants to put in...)

Modification:

**Stealth suit use** = True => Distance < 2 meters

Et cetera

*Activated rule:* CharmPotion(object)

If **CharmPotion** use = True

Then: **PlayerAvatar**; (Charm x 2)

While: distance **PlayerAvatar** – **ClassNPC** < 3 meters

AND

**useStart(CharmPotion)** <= 180 seconds

These are very crude examples but the syntax is easy to understand and with some simple grammatical rules implemented it could be learned in a very short time. The point is that this is not code that needs to work in an existing programming language, it does not need to execute, only describe what is intended using the vocabulary of the Granode method/tool.

It is quite possible that this could be created with a click and drag method.

4.5.3 Description with depth

Other aspects like background or semantic connection (see 2.4.2) that cannot be described as an attribute or capability can be linked to a database repository using either meta-tags (see 2.5.3) and/or hyperlinks.
4.6 Architecture & design of information

It is not known exactly how the architecture and design of the interface will look like when this paper is written since it is merely investigating what potential there is creating a desired tool with the knowledge and technology that already exists. Considering the similar applications mentioned in section 5.1.2 the available technology and methodology exists to grasp complex systems, it just has not been implemented in game design yet.

What is known is that there are some theoretical concepts that are desirable from a task oriented perspective but also from human psychology (which is credited in section 2.6)

4.6.1 General recommended features

After analyzing the shortcomings of the current GDD method, theories of human mental capabilities and existing features of software a preliminary list of what would be presumed desired features for a GDD-task specific tool includes:

- A person that does not know programming or graphic content generation should be able to start using it.
- It should have an easy to use interface where the game components, their attributes, their connections to other components and their behavior can be manipulated with a mouse as the main input method (see 2.6.1 & 2.6.3).
- A graphic interface (GUI) flexible enough to allow the user to find a layout that suits his/hers mentality. The user will also grow from a beginner to experienced user and should therefore have the option to streamline the personal workflow (see 2.6.1).
- The workflow should work from the core idea and outwards, meaning that the user should not struggle with the system to make something. Let the user make whatever he/she wants and let the internal logic of the connections (rules of the game) be the main concern. The system should not decide what can and cannot be, only if there is working logic present from the rules to support the connection (see 2.3.1-2.3.2 & 2.6.1, 2.7.1).
- Rule creation with simple logic operators from a predetermined list that applies to or derives data from any attribute, the rules should be easily accessible for later usage (see 4.5.2).
- There should be support for templates but they should not be a static framework that must be used (unless intended for specific export format).

51
The network grid is the primary interface, not the only one. When selecting which parts to be worked with in detail it should be possible to isolate that selection and open up suitable environments, either new window/workbench or a layer/filter function that isolates the selection (2.3, 2.5.3, 2.6.4).

These desirables are very vague and esoteric but at this time they cannot be more specific than this since the actual design is unknown at this point. The description for the more functional level should be done through information architecture praxis and the information design should be assembled with interaction design guidelines (see 2.6.1).

### 4.6.2 Recommended technical functions

In finding the maximal potential between the different theoretical methods and concepts a list of some desired features from an improved GDD perspective is compiled here:

- Every separate object or object type in the game (including the perceptible components of the world itself) should be represented by a graphic icon (a node) that is defined with as many attributes as necessary (see 2.3.1-2.3.2 & 2.6.2)
- The user should be able to create all the nodes from scratch and give them any desirable attributes free from prerequisites, any connection between attributes and/or rules should be possible as long as the syntax is valid (see 2.3.1-2.3.2, 2.6.1, 4.5.2)
- It should be possible to use graphic content in the description and/or visualization of the individual nodes. If there is a sketch available of a character it is helpful to use a thumbnail version of it on the node in the connection grid and any other occurrences where this entity is used (see 2.6.1 - 2.6.3).
- Any information that is not connected to something tangible or cannot be described through nodes should be available in a hyperlinked repository (see 2.4.2).

This list is not complete; there will be additions and edits during the development process.
4.6.3 **Other capabilities for Granode**

During the study it became apparent that the network grid that connects all the nodes is a form of object-relational database structure and could be used as such. As said in (2.7.4): “when creating the data in a relational database management system (RDBMS) a user creates lists of items and assigns them attributes with a value.” This sort of structure is generated as the objects/nodes are defined at creation and relationships between objects are defined.

This means that Granode could also be used as a Production Asset Management Software (see 2.5.5). It would be very useful for management purposes, everybody can see what has been done and what is slipping in schedule since all nodes can be attributed a status of completion marker.
5 Discussion

Goals without tools rarely exist, if there is a will there is a way; it is just a matter of finding a way/method/tool that works well enough to keep the will alive. Considering that computer games have been around for 30+ years now and they are becoming increasingly complex and numerous, there is much evidence that the will to make games is thriving.

Although this will to make games could easily be based on profit in larger companies just as it is a joy making games in smaller. Point is that the current ways of making games work well enough to not only give the byers more choice but the games apparently deliver what the paying customer wants.

But regardless if the GDD process works well enough or not, every game developer would benefit from a less arduous and a more effective documentation. And in section 4.6.3 some other benefits and/or opportunities that could build on this new method/tool were presented.

Technology is artificial constructs made by man (both physical and virtual) that help users achieve what would be impossible or to some extent inefficient without it. Any technology is invented with the sole purpose to assist human desires to be satisfied. If the desire is practical or for superficial entertainment is merely a label, regardless of label, desires evolve. This is particularly true for games and game makers.

The technology to manipulate what we see on a screen is mature enough to make it possible to guide a programs attention and user intent on a screen with touch or click alone. It is possible to see abstract concepts like growth or time as geometrical shapes. Navigation is illustrated with images instead of written instructions. This behavior is designed by humans to let users communicate with each other more efficient and with computers to make them do what the user wants easier and more intuitive. It is not inconceivable to ask of something similar from human to human communication in game design.
5.1 Respect for the craft

“Creativity is really heavily policed by rationality, whereas rationality actually goes dangerously un-policed [...] If you think creativity is expensive you should try logic. Logic can be spectacularly expensive because it takes you down a line of pretty much unquestioned assumption, without ever asking the question: is this whole thing based on a completely erroneous premise?”

Rory Sutherland, Talk - business of software 2011

One aspect of what Mr. Sutherland conveys in the quote above is that one must dare to be creative, taking a leap of faith and give room for play and experimentation in order to come up with “the next thing”. It could also apply to the idea of taking rational ideas into the human domain to discover that humans are not rational. This applies very much to games and even if “making games” might sound flimsily it is hard, many try but most fail; those who survive learn quickly or perish.

Even if the object of the game is to entertain through bizarre concepts it must still work on real hardware, guided by actual logic and be sold with heartless economics. People with experience in game production are good at what they do because making games is a qualitative process, not quantitative. The two main qualities that are sought after are the ability to work in a game production environment and being good at using the tools of the trade.

5.1.1 Working in a game production environment

The term “game production environment” sounds frustratingly generic to many who have worked in more than one game company because it is special everywhere. In fact the common denominators in “the industry” as a whole are the type of products it produces, how the details is made/done and the mentality required. It is not all different but hierarchy, workflow and structure comes in different formats and combinations. The mentality of a person that works with games should be ready for sudden changes both on the specifics and on the grander scale, these become scarcer with time as production progresses but it is difficult working conditions for some and they either adapt or leave.
5.1.2 Tools of the trade

Being good with a tool has many measurable qualities that are weighted in when assessing usefulness but there is one that is most important; speed. If a person is fast he/she has the time to be detailed, diverse, thorough et cetera. Having a fast workflow is key for being useful enough but if the person is not interested in growing by starting using additional/new methods and sub-tools he/she will not advance much in larger organizations since they upgrade all that is necessary to stay competitive.

Graphic content generating (and some audio) software is annually upgraded with newer and better functions and the results is improving with the increased capabilities. Programming software is continuously updated with smaller additional sub-tools since the basic premise of programming is the programming language itself and that does not change over time. Game designers get new tools with every release of the office package (or equivalent), which is about once every three to four years and the core concepts have not changed much the last fifteen years.

Going back to Mr. Sutherlands quote, is this a “line of pretty much unquestioned assumption”? Are game designers destined to use word processors and some charts as the only available tools?
5.2 Validity

There is no outspoken need to make Granode, but when searching the literature and interviewing people in the business there is a substantial amount of friction coming from this stage in the production. It is deemed important to improve this process since it lies early in the overall process and many subsequent decisions are based upon the information contained.

The people interviewed have had relevant knowledge in either game design, programming in a game design environment and/or management skills for large organizations and they all agree that this should cause major problems in any other industry. It is very unlikely that the gaming industry is immune to these effects; it is more probable that it is to some extent less sensitive than others. Even if it is only less sensitive, improvements should make the overall process more efficient.

5.2.1 Credibility

There has been no claim that contradicts the mainstream belief within the industry. All the similar examples mentioned comes from well-established sources that have proven that the concept works very well in that context. Everyone interviewed has had direct involvement in the areas investigated and have significant credibility in their field.

The inherent ad-hock nature and diversity of the industry gives a somewhat feeble impression both in and in comparison with older “serious business” but the revenue data speaks its rational language, digital games is growing rapidly making a lot of money and show no indications slowing down.
5.3 Conclusion

When designing games there is an established need to keep extensive and continuous documentation available so that everybody in the project can follow the progress and get the latest updates. Unfortunately this process becomes too extensive and cumbersome rather quickly and even if the documentation continues it has no readers since it is very unlikely that any old information is updated and any new information is reliable. The general mentality about this issue seems to be that the process is sometimes painful and full of inadequacies but it works well enough to leave alone. So the original question asks:

Is it viable to replace the text-based descriptions of the interactions and capabilities of proxies in a GDD with an interactive node-based graphic navigation interface and maintain equal or improved functionality?

The core of this problem is that the amount of text that needs to be written is too abundant for the people that are assigned to do it. Granode cannot reduce the amount of actual description of a game but hopefully the amount of work and time to do it by using a different method and taking consideration to who is supposed to read it and how it is presented to them.

A comparative analysis of several software packages indicate that the type of structure needed to rectify this problem is available and could be implemented to the specifics of a perceived distinct industry. This structure could be rendered through a tool that requires no specific programming skills or other high-end software package that is usually required for the industry but still much more versatile than the word processor that is used today.
5.4 Continued research

It is proven that it is possible and potentially significant to implement a new method and develop a new tool and the actual technology and individual methods already exists, it just needs to be put together in a configuration that suits the game industry.

This paper suggests that a new software tool could be created to address many of the inherent problems that exist with today’s methods. This tool should be researched and prototyped to validate the theoretical claims expressed here.

It is also possible to build upon this software to create additional modules that could extend specific functionality which would allow more powerful and specialized tools if they are required. This could be done with an (at least partially) open source format to create a community which has been proven a powerful and versatile development method, if successfully implemented.
References

Books:

Cooper, A (2004), *The Inmates Are Running The Asylum*, SAMS, Indianapolis IN


Dille & Platten, F & Platten, J (2007) *The ultimate guide to Video Game Writing and design*, Lone Eagle Publishing, USA


Weinschenk, S (2011) *100 Things every designer needs to know about people*, New Raiders, Berkeley CA
Articles:
Callele, D & Neufeld, E & Schneider, K (2004), *Requirements Engineering and the Creative Process in the Video Game Industry*, Department of Computer Science University of Saskatchewan Saskatoon, Saskatchewan


Digital source:
Gamasutra.com – The art and business of making games

Urbandictionary.com (Swiss army knife syndrome 2012)
Retrieved: May 24, 2012


On-line video:
Shirky, C (2009), talk on *Web2.0 Expo NY*, Retrieved: May 18 - 2012

Wikipedia.org (All information retrieved 28 May -2012)

Wikipedia.org (Data base 2012)
This page was last modified on 22 May 2012 at 04:29

Wikipedia.org (DIM 2012)
This page was last modified on 24 February 2012 at 11:44

Wikipedia.org (Flowchart 2012)
http://en.wikipedia.org/wiki/Flowcharts
This page was last modified on 3 June 2012 at 01:44

Wikipedia.org (Game Companies 2012)
This page was last modified on 23 May 2012 at 22:25

Wikipedia.org (Graph-theory 2012)
http://en.wikipedia.org/wiki/Graph_theory
This page was last modified on 23 May 2012 at 05:04

Wikipedia.org (Information design 2012)
http://en.wikipedia.org/wiki/Information_design
This page was last modified on 15 May 2012 at 19:20

Wikipedia.org (meta-tag 2012)
http://en.wikipedia.org/wiki/Tag_(metadata)
This page was last modified on 12 May 2012 at 01:04

Wikipedia.org (ontology 2012)
http://en.wikipedia.org/wiki/Ontology_(information_science)
This page was last modified on 29 April 2012 at 00:54

Wikipedia.org (Programming 2012)
This page was last modified on 17 May 2012 at 09:26

Wikipedia.org (Specification language 2012)