MADMAN (media distrubtion management)

Systems Requirements Specification

Version 1.0

About This Document

Purpose of this Document	The Systems Requirements Specification (SRS) is designed to express the behavioral, performance, and development requirements of this product and serves as the fundamental requirements document for the development of the product. The Systems Requirements Specification includes a description of every input into the system, every output from the system and all functions performed by the system in response to input or in support of an output. The SRS meets IEEE830 standards and is the exclusive requirements document to be used in development; all design and testing choices must be compatible with this document.
Document Prepared for	MADMAN Developers and Interested Developers and/or Users of the project.
Intended Audience	Developers, Clients. This document is mainteined to: keep the developers sane and organized during the projects development lifecycle, and also provide a way for interested parties to become familiar with the project if interested in committing code.
Date of Publication	06/26/10
Prepared by	Chris Sherman for Project MADMAN
Copyright Notice	© Madman

Revision History

Date	Revision	Description	Author
06/19/10	1.1	Initial documentation	Chris
06/20/10	1.2	Finishing Initial Documentation	Chris
06/22/10	1.3	Refining overall scope of project	Chris
06/26/10	1.4	Defining functionality for versions of the project	Chris

Table of Contents

1.Introduction

1.1.Purpose

The Systems Requirements Specification (SRS) is designed to express the behavioral, performance, and developmental requirements and serves as the fundamental requirements document for the development of MADMAN. The Systems Requirements Specification includes a description of every input into the system, every output from the system and all functions performed by the system in response to input or in support of an output. The SRS meets IEEE830 standards and is the exclusive requirements document to be used in development; all design and testing choices must be compatible with this document.

1.2.Scope

The scope of the SRS identifies the magnitude of what the product will cover. The Systems Requirements Specification is intended for review by all parties interested.

Readable: minimizes misinterpretations by developers and testers by using clear, concise, unambiguous, and complete language. This was everyone is communicating in a common vocabulary about the project.

Testable: will be used for strategies in Master Test Plan, Functional Test Plan, and other formal testing documents.

Modifiable: changes easily as requirements change, accommodates Requirements Traceability Matrix, see section 4 of this document, and Change Management Policy.

Useable for Operations: will be used as a living document after project iterations for deployment, operations and modifications. Additionally, will serve as a primer for interested developers to obtain a greater understanding of the system when changes need to be made or features added.

Traceable: works with the Requirements to ensure components, responsibilities, and test cases link to minimize impact for requirements change and localize defects.

Buildable: will be used by developers to write code from specifications.

Documentable: will be used to write user manual and technical manual to help acquaint new clients, interested developers and potential parties.

1.3.References

Http://www.imdb.com — API for movie information

Name	Purpose	Link
Internet Movie Database	Movie item information	Http://www.imdb.com
Television Database	Television item information	Http://www.thetvdb.com
	Music database	
	Game database	

1.4. Standards

IEEE 830-1993 – The content and qualities of a good Systems Requirements Specification (SRS) are described and several sample SRS outlines are presented. This recommended practice is aimed at specifying requirements of software to be developed but also can be applied to assist in the selection of in-house and commercial software products.

1.5.Definitions

Term	Definition
Library	
User	
Admin User	
Power User	
Regular User	
Item	
Media	Including games, applications, movies, music, telecvision episodes
Madman	Media distribution and management

Scraping	

2.Overall Description

This section of the SRS describes the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in section 3, and makes them easier to understand.

2.1.Project Abstract

Project Name: MADMAN

Authors: Mark Arnold, Chris Sherman

2.1.1.Project Scope

Madman's intent is to reduce media management and streamline content distribution. The hope is that MADMAN will be the sole solution for management of media and provide an easy distribution system with built in user acess control which will be flexible yet resilient to corruption.

2.1.2.Background

MADMAN derived from an ambitious attempt at streamlining media access remotely and locally. The idea was simple and broad in nature, however, to achieve or implement the desired result is an extensive task. Mark had a lot of media, really no lie and his dream was to provide a way for himself to easily access this content anywhere he wished. Additionally, he wanted to provide a way for friends to be able to easily and seamlessly access his content. To achieve this he started Raptorwire which is a web interface provided a view to clients (himself and his friends) which yields the abiility to easily filter or sift through his media and access anything at anytime. Along the way various additional pieces and components were added and the end result was a quite extensive project which jump started our dream to create the ultimate media management and distrubution solution. This is how project

MADMAN started, from an idea, to the ultimate all inclusive media distribution and management implementation.

2.1.3.System Purpose

Who — Clients of MADMAN will be any party interested in consolidating their media.
Where — The system's physical location varies, but the user interface will be remotely accessible.
What — The system will provide user accounts and incorporate access control for media content.
Why — To consolidate all media and provide a interface to access said content.

2.1.4.System Mission

MADMAN's ultimate mission is to consolidate client's media and provide them with an easy to use interface to manage it. Additionally, MADMAN will provide the user with a built in distribution system providing ability to share and stream content to remote parties via HTTP protocol.

2.1.5.System Functions / Responsibilities (make these measurable somehow)

The system will be implemented in a agile and iterative fashion. Therefore, functions and responsbilities will be define per version with each version or revision of the system expanding on the prior.

V0.1 — Initial database build.

The system will perform initialization of the project. It will use a configuration file to understand root locations for media content and read through these locations to populate the database with file entries.

At this point the database will have all of the file information and can begin scraping against online databases for additional information pertaining to each piece of media.

2.2.Functional Objectives

> [<Priority> The system shall <function>, creating <IR AC IS benefit>. Evidence of success will be measured by <x> improvement in <measurable performance>]

2.3.System Constraints

System Constraints restrict options of design, behavior, appearance or operation. They become requirements due to factors outside the normal problem domain. System Constraints describe how the product operates inside various circumstances and limit the options designers have if building the product. This section specifies design constraints imposed by other standards, hardware limitations, communication interface limitations, etc. There are a number of attributes of software that can serve as requirements.

2.3.1.User Interface Constraints

- [v0.1] It will take a user less than one hour to become familliar with general media navigation and basic functionality. The web interface for Madman will be intuitive and easy to understand. A user should be able to easily become familiar with how to navigate the UI.

- [v0.1] It will take an administrator or power user less than one hour to become acquanted with the administration activities the site provides.

- [v0.1] It will take less than thirty minutes of user intervention to populate the and initialize the project. (Although it must be noted that this will take significant time for the system to scan large libraries.)

-

2.3.2.Hardware Constraints

Our intention for MADMAN is to not constrain the user to hardware. However, it should be noted that there are considerations to make in regards to using the project in terms of your hardware availabilities.

Storage Consideration

A clients deployment of MADMAN will be limited in storage to the available hard drive space provided for the project, as developers our estimation for our own deployment is 10 Terabytes easy but you may need much more it all boils down to how much media you have.

Speed Consideration

The server is in some situations only as powerful as its CPU and for that reason obviously a fast CPU is desired but any current x86 or x64 processor should suffice. CPU considerations

need be addressed if said client has intentions of sharing content remotely with users as MADMAN will make every effort to compress packages prior to sending and compression is CPU intensive. However, most of MADMAN is I/O bound activity because the idea of MADMAN is to distribute and provide easy access to content. Therefore, SATA will be faster than IDE and if you have the money SSD will be faster than both.

Furthermore, as in most hardware considerations the more RAM the better. We recomend 2GB but are confident on a small deployment (small in terms of number of users) that 512MB or 1GB would be sufficient.

2.3.3.Software Constraints

Software constraints are fairly straight forward although they are not constraints as much as recomendations. Development has been done on linux based machines using PHP, MySQL, Perl, and Python. Aside from Linux all of these are available on a windows platform and for this reason there should be no problem with deploying MADMAN on a windows based system although we cannot assert correctness.

2.3.4.Communications Constraints

The system must be networked properly but this depends on your desired deployment. If client desires a local deployment then any client of the MADMAN deployment must be able to reach the server via local area network. For remote access, the LAN must be configured to provide access to the server in which MADMAN is deployed on.

Additionally, the server should be able to have direct access to the location of clients storage. This can be achieved in different ways but recomendation is to use a samba share, sshfs, or direct connection to the server.

2.3.5.Data Management Constraints

The system must have local or remote access to the database where MADMAN will store its model. Additionally, internet access will be required for media scraping.

2.3.6.Operational Constraints

Below is a list of restrictions on how the product will run when in its environment.

For example,

The throughput of the system shall be kept to x seconds on average but to a maximum of y seconds if the system is under heavy load.
The throughput of the system shall be y transactions per second.
The system shall have an average of 10 current projects stored and up to a Maximum 50 projects.
The system shall have an average of 50 workers currently logged in and up to a Maximum of 100 users logged in.
The system shall use a maximum of x% of memory, y% of disk space and z% of company internet bandwidth.
The system shall be available x% of the work day which is between 6am and 6pm.
The system will have x hours of Mean Time Between Failures (MTBF).
The system will have y hours of Mean Time To Repair (MTTR).
Maximum Bugs or Defect Rate
Minor Bugs (displaying incorrect information, other display glitches) these will be kept to a maximum of x bugs/KLOC.
Critical Bugs (such as user having too much access, not having enough access) these will be kept to a maximum of y bugs/KLOC.
Major Bugs (such as system crash, loss of data, failure to create output) these will be kept to a maximum of z bugs/KLOC.

2.3.7.Site Adaptation Constraints

Currently, MADMAN during installation requires the user have specific directory structures for installation and initial setup. These are outlined below:

These are under the assumption that

Movies

Music

Games

Applications -

TV — TV Show Name / Season # / File named as Episode Number.<media extension>

2.3.8.Design Standards Compliance

There are a number of attributes of software that can serve as requirements. The following list specifies factors that must be established for the system to work. Existing standards and regulations impose the following constraints on the system.

For example,

The system shall be designed using open sources languages and or freeware software.
The system interface shall be developed using the Java Programming Language version 1.5 to be able to run on multiple OS.
The system shall be developed using MySQL version 5.0 or higher to store data.
The system shall be developed using CLIPS version 6.24 to help build the expert system.
The system shall be designed be able to switch interfaces and or back end (databases).
Front End software should be designed for easy portability to Aspect J or AJAX software.
Back End software should be designed for easy portability to a higher capacity and faster database system.System Operational Parameters

2.4.Other Constraints

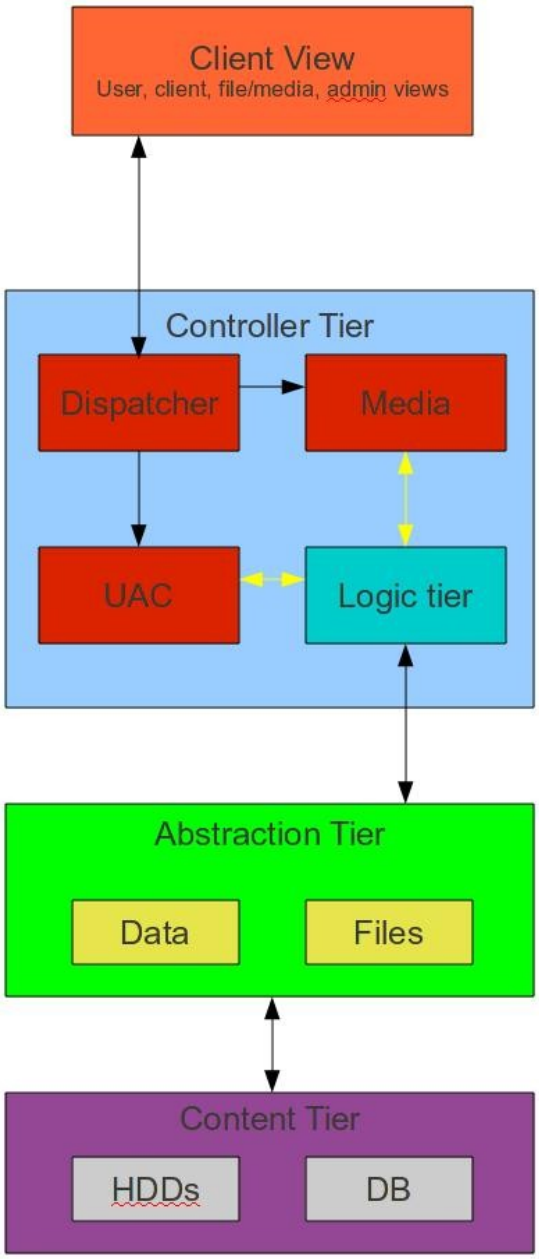
Below is a list of all other interfaces that impose design constraints.

3.System Events and Data Flows

3.1.Event Table

[Occurrences at a specific time and place that trigger system processing]	[data inflow or time that system detects]	[ultimate creator of trigger. May be a person, department, or system. If event type is temporal, this is left blank.]	[system process that results from trigger]	[data that system produces. If only internal effects are made, then this is ‘n/a’]	[ultimate destination of data response.]

3.2.Context Diagram



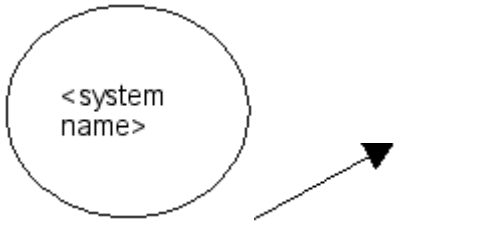
Context diagrams use data flow diagramming (DFD) notation to illustrate the scope of a problem and the source, sinks of data and control that flows into and out of a system.

<external: data source or destination>

<inflow: group data item>

<inflow: group data item>

<external: data source or destination>



<external: data source or destination>

<outflow: group data item>

<outflow: group data item>

3.3.Product Functions - System Activities

This subsection of the SRS provides a summary of the major processes that the software will perform, which includes the system tasks and features from the Product Requirements document and Project Charter.

[Activity]

[Description]

<repeat for all activities>

3.4.User Characteristics

User Characteristics describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise.

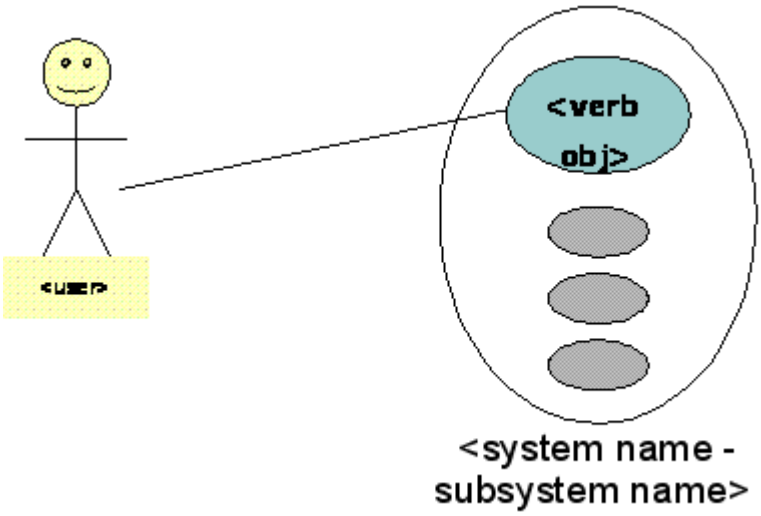
<system name> users consist of the following:

Managers who wish to perform system administration functions as well as export company financial information.
<repeat for all users and system externals>

4.Specific Requirements

This section of the SRS contains all the system requirements to a level of detail sufficient to enable designers to design a system that satisfies those requirements. Testers can use this section to test that the system satisfies those requirements and technical writers can create the necessary support documentation for operations and maintenance. Note: Use Cases are in priority order.

4.1.Use Case Diagram - organized by subsystem



4.2.Use Cases

Use Cases are requirements from the Client translated into unambiguous language. A Use Case may have multiple inputs or outputs as part of the same functional flow. A Use Case without any input or output is not valid. The detailed requirements of a Use Case tend to be extensive. For this reason, it is recommended that careful consideration be given to organizing the requirements in a manner optimal for understanding. Subcases are identical to use cases except where noted. This section provides descriptions of all the use cases devised for this system.

Each use case description provides the following information:

4.2.1.Use Case Scenario <#>

	<Use Case Name>
Purpose	A brief description of what the user is trying to accomplish.
Actor	A person or external system outside the scope of the system that triggers step one of the Detailed Description.
Input Data	A list of all external data needed for the use case to be performed.

Output Data	A list of all data produced by the use case execution.
Invariants	A condition which is maintained throughout the use case. This section is used to highlight assumptions made for the sake of the use case.
Pre-conditions	Conditions which must hold for the use case to be applicable. It is assumed that these conditions are true prior to the beginning of the use case, and will not be true when the use case completes.
Post-conditions	Conditions which are guaranteed to hold after completion of the use case.
Basic Flow:	A single, error-free path, which may contain subflows, calculations, logical structures, etc.
Alternative Flow(s):	All exception and error cases, including where/how they were triggered
Extension Points:	<<includes>> and <<extends>> cases and where they were referenced
Business Rules:	The rationale for this case, also explains exceptions and errors
Notes	Any other relevant information not included in the above sections.

4.2.2.Use Case <#> Prototype

[Complete set of simple discovery prototypes showing all user interaction for basic and alternate flows.]

4.2.3.Use Case <#> Object Interaction Diagram

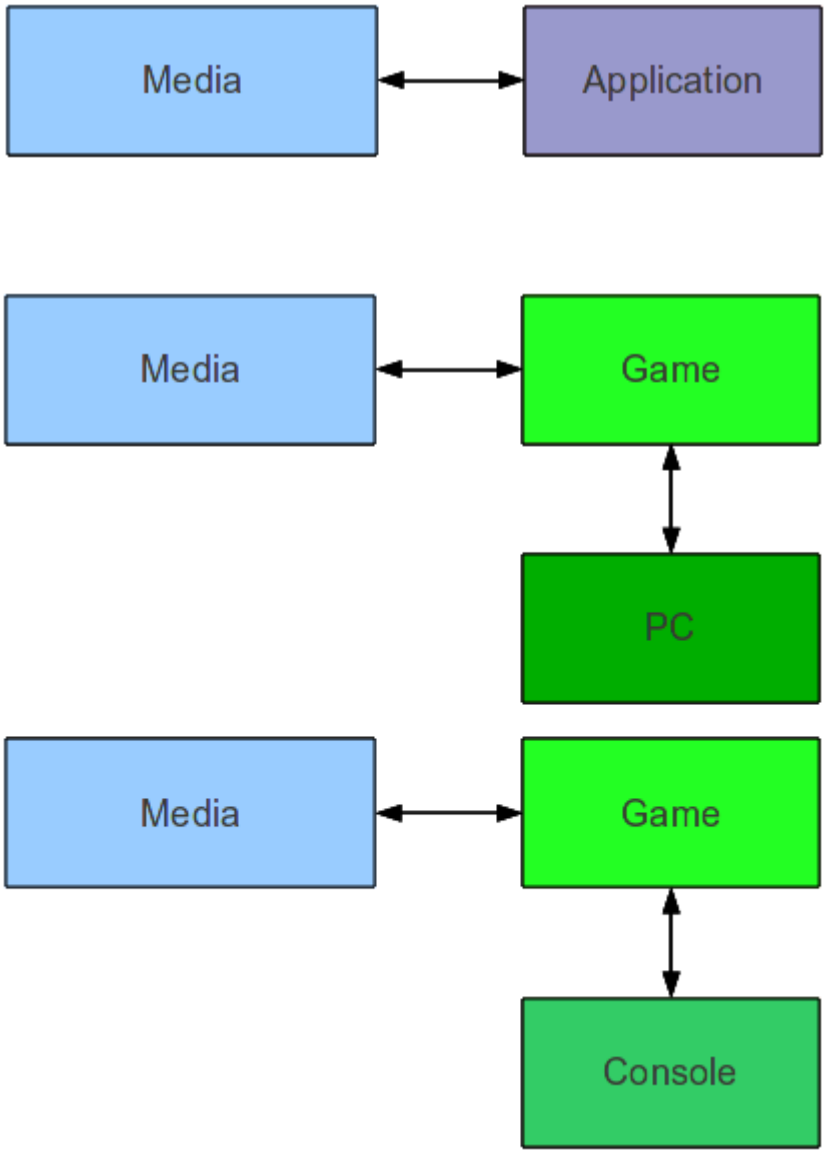
[Sequence or Collaboration diagram showing all participating classes and messages that trigger response for basic and alternate flows.]

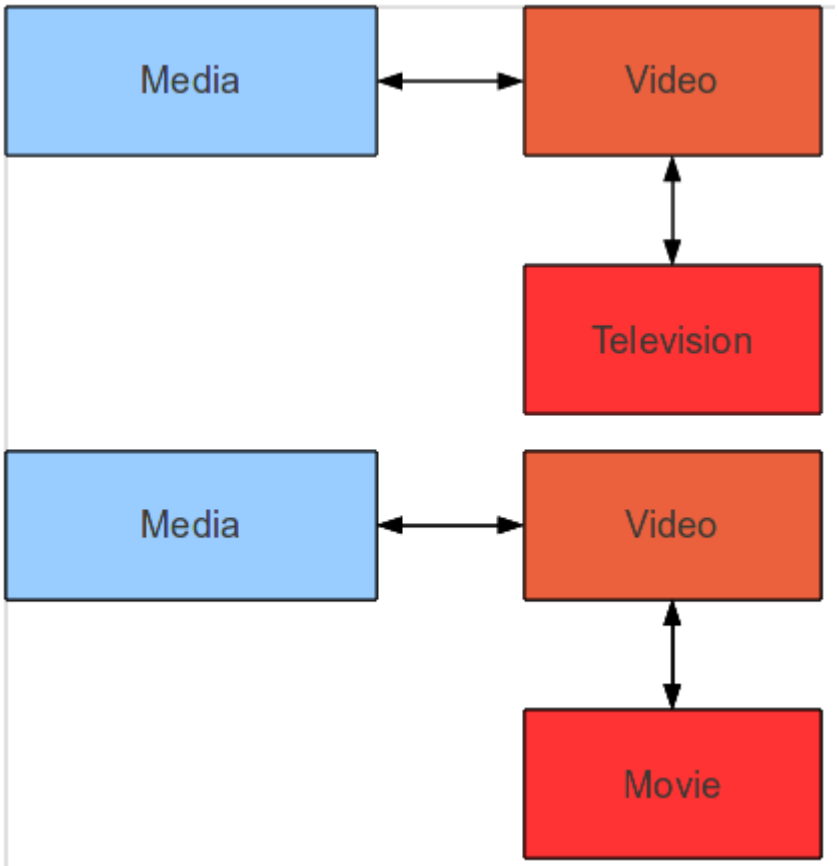
<… repeat for all Use Cases…>

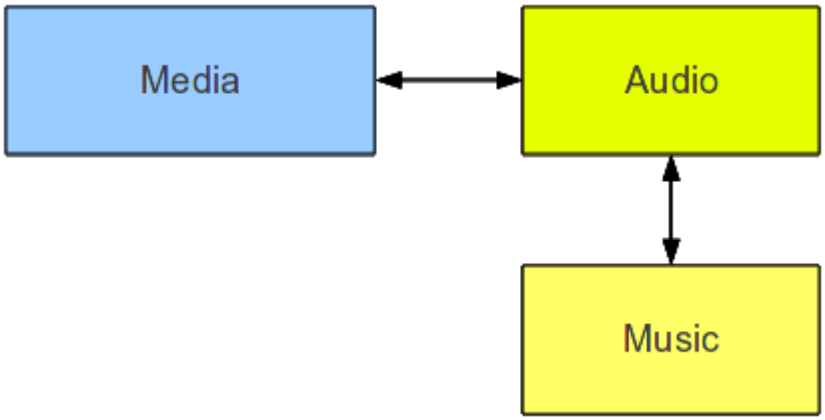
5.Validated Object Model

The Validated Object Model is a visual representation of the idealized problem domain. The consistency between the Sequence Diagrams and the Object model validates the requirements.

5.1.Class Diagram




Media
Video
Television
Media
Video
Movie


Media
Audio
Music

5.2. Class Specifications

Class	Madman.Media
Parent	null
Description	The base class for all types of media.
Attributes	
Methods	

Class	Madman.Video
Parent	Media
Description	Class to extend media, specific to video content.
Attributes	
Methods	

Class	Madman.Movie
Parent	Video
Description	Class to extend media, specific to movie content.
Attributes	
Methods	

Class	Madman.Audio
Parent	Media
Description	Class which extends media and is specific to audio media types.
Attributes	
Methods	

Class	Madman.Music
Parent	Audio
Description	Class which extends audio and is specific to music media types. Such as albums and songs.
Attributes	

Methods	

Class	Madman.Television
Parent	Video
Description	Extends media class, with more specific methods applying to video content.
Attributes	
Methods	

Class	Madman.Game
Parent	Media
Description	Class to extend media, specific to game content.
Attributes	

Methods	

Class	Madman.PC
Parent	Game
Description	Extends game, with methods applying to PC games.
Attributes	
Methods	

Class	Madman.Console
Parent	Game
Description	Extending game class, with methods applying to console video games.
Attributes	
Methods	

Class	Madman.Application
Parent	Media
Description	Methods applying to applications, specific to pc applications.
Attributes	
Methods	