MOG Factsheet:

- Officially released in Dec. 2006
- MOG is a client/server software solution designed specifically for the video games industry and the consoles they develop for. A solution with years of gaming experience behind it, MOG provides a safer, faster pipeline for making better console games.

MOG Q&A

Does MOG compete with existing SCMs such as Perforce, Subversion, or CVS?

- MOG is much more than just a traditional SCM because it takes on a lot more elements of game development. Yes, there are some overlaps because MOG does support version control for data but it goes a lot further in addressing the team's need to automate content creation workflow.
- MOG was built specifically for data and is not intended to be used for code. Teams that adopt MOG will still need to have their programmers using their SCM of choice.

What is MOG?

- MOG is a Windows application created to facilitate safer and faster development of better games on all gaming platforms.

How does MOG make it safer?

- **Version Control** – Unique to MOG, we revision asset containers which can be made up of one or more game files. A prime example for this is a complex asset like a level that consists of multiple files. Programmer-centric version control systems would revision each file separately. MOG revisions these as one single asset, making it easier and safer to roll between revisions. This feature is just one reason why MOG is the best solution for managing game content.
- **Branching** – A full product branch, which means we branch both the source assets along with their exported and processed counterparts.
- **Inheritable Properties –** On setup of the project, asset classifications can be created that define how content will be processed into its final form. Subsequently, any non-technical person can then freely add new assets simply by choosing the correct classification. All the correct pre-set options and conversion settings will be applied under the hood. For instance, the artist only needs to know that a texture they are using is an alien character texture, and MOG will automatically convert it with the inherited alien character conversion routines and package it into the appropriate alien packages.
- **Automation** – Assets get converted and handled exactly the same way every single time, regardless of time, talent, external pressures, fatigue, experience or any other factors that can be attributed to human error.
- **Flow Control** – Because so much content can be created or modified in a single day, MOG uses its flow control to provide a safety net for pending milestones or deliverables. With a few clicks, all content can be rerouted to a team member or QA department. At that point, the content can be individually tested and verified before being promoted to the final milestone build.

How does MOG make it faster?

- **One-Click Deployment of Content** – With the proper integration between a game and MOG's command line, MOG is a one-click export to target platforms.

- **Simultaneous Multi-Platform Development** – Using MOG's distributive ripping (like a render farm), asset conversions are performed simultaneously for all target platforms without it costing any additional time. Having multiple platform-targeted workspaces allows users to conveniently switch between platform builds with the click of a button.
- **Custom Tool Box** – A workspace-specific set of tool accelerators designed to give quick access to platform tools and scripts. For instance, a "run game" button can be created that will run the workspace-relative game executable. There are much more advanced examples, but this ensures that content creators are working with the correct tools and data tree at all times with no extra effort on their part.
- **Distributive Ripping** – The render farm for digital asset conversion, distributive ripping takes advantage of CPU power from the entire network, making it possible to process thousands and thousands of assets much faster than a single machine.
- **Use Existing Tools** – We don't force studios to adopt new tools. Not only do we embrace existing tools, we provide an extensible automatable framework that gives them the power to do it their way with the speed, power and safety of doing it our way.
- **Atomic Asset Conversion** – To see an asset on the target platform, we only need to wait for that one asset to be converted, rather than the entire build process to complete for the desired platform. Atomic Asset Conversion makes for a very fast workflow as well as a more surgical approach by guaranteeing that nothing else is affected by the change.
- **Automation** – Because MOG has embraced automation, performing conversions on a massive scale can be done effortlessly. For instance, MOG can convert thousands of textures in our distributive ripping system from a PSD to a DDS in minutes. (http://www.mogware.com/product/performance.html)

How does MOG make games better?

- **Creative Collaboration –**Modern games are made up of interdependencies, with groups of artists working together to achieve a common goal, rather than one person doing it all. MOG facilitates the sharing of assets within a team without necessarily using the standard check-in/check-out paradigm used by every other content management system. To be more specific, we still embrace the typical check-in/check-out paradigm, but have another method that specifically enhances team collaboration. We call this method "**Lateral Asset Sharing.**"

  A simple example of L.A.S. is with a character for a game. The modeling, rigging, texture mapping, skin binding, and animation will likely be done by different people. The modeler might change the model in a way that requires interaction and collaboration with other team members. MOG's interface makes it easy and safe for team members to work together without any of the complexities often found with this level of team collaboration. Artists can experiment, test out, and share assets with each other without causing potential instabilities that often spill over and affect the entire team. This process means that games, through better collaboration methods, can become more creative and more fun.

  In situations where packages are involved, such as with Unreal Engine 3 technology, it is impossible for team collaboration at this level. One user can check out a package, locking all other team members out. Once his work is done, he checks the entire package back in so the next user can begin their work. That cycle continues until the package is complete, which is very linear and time-consuming. With MOG, all the different parts of a package can be worked on simultaneously, each person working within the package at the same time in a safe and efficient way. Team members no longer need to wait in line for another user's lock to be released on a package.

We believe that great games are discovered, not engineered. Allowing single-user experimentation is essential for any pipeline to discover creative elements, but allowing multiple users to share on collaborative components makes a good pipeline great. It is this level of collaboration that fuels the creative forces of team members, allowing them to discover what makes their game better.

- **Simultaneous Multi-Platform Development –** Consistency of assets is much greater because all platform files are processed from a single export. When developing for 3 platforms, no single platform will ever fall out of sync because of the way MOG supports true simultaneous multi-platform development. Many game engines have created a situation where you finish the PC version first then port the packages to the other platforms. This is a difficult and time consuming approach because problems are almost guaranteed to come up in this process. Attributes and requirements are unique to the various game consoles. The best time to encounter issues is not on the last day of a milestone deadline. With MOG, you can track and test this all along the process of development so you can be confident of the status of your game at any stage of development. This also means that artists can deal with content issues while that specific content is still fresh in their mind. Revisiting year-old content is inefficient and risky, so we try to minimize the need for that. Export once and allow MOG to do the rest.

Automation

  - Automation facilitates rapid iterations, which is key to making content better. Making it easy for artists to see their changes quickly encourages experimentation with the assets and ultimately leads them to discover what will make it better.
  - It facilitates mass asset optimization conversions. It is these optimizations that are required in order to make the game run faster, play better, load quicker, look better, etc.
  - "If it's monotonous, no creative person should ever do it." Over 80% of a modern team consists of creative people. If you make creative people do monotonous things, they become less creative and you end up wasting a lot of time and money.