# machinesForLivingManifesto

2012-10-02 ~ Theo Armour ~ first rough draft

### Preface

What follows is a skeleton description of an app followed by an unorganized list of further specification items. Most every item in this document is based on existing code or practices but have not been used previoysly for the described purpose

This paper has at least a week to go before it gets to be interesting but, at least, it’s a start.

### The App

It’s actually a new way of designing. On the other hand it’s as unobtrusive as DropBox. It just works. There’s nothing to learn.

What does the app do?

The server-side app looks at the work being produced for a design and notifies the responsible parties of any errors and inconsistencies in the project.

The app is a system of pulling and pushing, forking and merging design data (and code) much in the same way that git and gitHub do for programming code.

The app covers transporting copies of local data to servers. Once design data is on the server it is checked in a variety of ways. See a paper titled “CAD Lint”

At any time the app can be called to display multiple version of the same thing. These could be different revisions created by the same program or the same thing as created by completely different programs such as Revit and SketchUp.

On the simplest level, because the two (or more) objects are occupying exactly the same space that any significant differences will be immediately and outwardly visible. The app may also be able to highlight discrepancies via differences in the placements of vertices, faces bounding boxes of elements.

Premium versions of the app might offer the ability to creating script that would update the models so as t resolve the issues.

All of this should be familiar to anybody who uses git or any of the popular revision control system

You can use VIm, Eclipse or Visual Studio, it makes no difference. What the app does is to ensure that the code created by whatever designs tools is logical, usable and can be merged into the core code stream.

### Continuous Revision Control

Architects, designers, and engineers tend to use drawing approval applications that are proprietary to the same vendor making their CAD application. These programs also cover drawing back up

Designers and engineers need GitHub or something like it just as much as coders do. This includes the ability to check for differences and then merge them.

### Two Types of Revision Control

The design process is under revision control.

The product/thing/building being built/manufactured is under its own revision control.

The two different revision control systems should be able to live with each other.

### Seamless sync between cloud and desktop

Design files must no longer be the prisoners of hard disks in desktop computers. Designers need something like Camino dropbox folders where the file is locally based but also in the cloud and on other computers the same time.

The engineer or designer need do nothing to make this happen. Once installed and setup, the app takes care of the classification all by itself. It just works.

### Ever-present 3D UX/GUI

We don't really need or want two separate user interfaces. If we are going to design and collaborate in 3-D, then we need to stay in 3-D the whole time

### Reactive / always animated / behaving /

In real life are designs go from grass covered plots to thirty story buildings in a process taking years . We should be able to see what the site looks in week 30 of construction and also five years after it has been built when it begins to need a renovation.

### Augmented unreality - go on site and see the design

We want to be able to walk around a building site and see what our design would look like using our smart phone.

### Content/Style/Behavior Split

In designing websites we have a split between HTML/CSS/JavaScript . These cover content, style and behavior. The design professions need the same kind of separation of functions

Code viewer/editor - online and desktop

- not a table/database viewer

### MMORPD

We need massively multiplayer online role-playing design. A design team for a large project that has have dozens of specialty professions applying the skills to the project. The same kind of thing happens with guilds in a large online game.

### All formats accepted

“Out tech guy built an app, which is actually quite cool, which uses a special file format. Can we keep those files?”

“We had this really old cad app that we really like but no really knows about it. Can we keep those files?”

I've been using AutoCAD since the beginning of time, you will have it drag my DWG files out of my cold dead hands.”

The system must be able to keep any and all types of files

### All files stored forever

You like to use Revit. She likes to use SketchUp. Fine. Whatever you use we will keep copies of every file you make.

### Everything has a URL

On your hard disk, big models use external references from all over the place in some kind of higgledy-piggledy manner . If that's how you choose to run your office, that's okay. But when we get the files online each file will have its own URL. Before we give the file a URL we will do a CRC on the file and see if we have a duplicate on our servers.

### Everything accessible via a RESTful API

Not only will each library block be accessible via a URL, aps will also be able to edit the files using a restful API

### Seamless cross-server support

You want your doors to be stored on the window manufactures website. And your doors to be coming from the service of the people who made the doors. That way, in case there is a change in the delivery schedule or an update to the specification, everybody will know about it as soon as possible . As long as the data is in some kind of JSON format for all very easy for us

Scale and timeline and physics standards

FOSS - like git, linux, apache, php

Enterprise usage

Post-Processing - apply codes/specs

Designs will be edited by robots

Ontology

3D timeline

Free/no cost

No Learning Curve