1.1 Welcome

TODO Frontispiece

Still on the lookout for a quote that befit the thesis.

TODO Abstract

Software is a continuous decision making process.

Things to mention:

- What is my unique proposition?
- Reasons for creation
- Research Aims
- Current Problems under investigation.
- Results and Implications.
- Methods and Procedures

Aims

Architect and engineer a modular ecosystem of applications thatform a unified platform to aid student learning.

Objectives

My objectives are four-fold with a re-orientation step in thisproject:

- To research and summarise the current systems in use and analyse their advantages and disadvantages and retain all attributes that are deemed to be desirable.
- To investigate about befitting architectures for digital ecosystems and on fostering communities around them.
- Iteratively develop the applications and devise an appropriate architecture for interoperability between them.
- To evaluate my design decisions based on feedback from colleagues and observation conducted on prototypes through focus groups.
- Feed the data obtained into the first step and re-orient myself in the light of the new data obtained.

Rationale

A university student has a plethora of digital tools at theirdisposal to stay on top of their academics. Even when these aretailored to enable the student to accomplish their tasks, lack ofan overarching architecture and conceptual

integrity. The lack ofharmony across metaphors employed, knowledge representation,consistent behaviour and visual language used for when interactingwith the user results in a chaotic platform which often works tothe detriment of the student. This often results in confusion, lackin productivity and erroneous decisions when trying to interfacewith these tools.

The hypothesis of this dissertation is to construct an ecosystemfor student learning that is architected from the ground up to forma cogent platform that students can leverage. This would be a setof apps that aid them in their learning process by hoping topresent coherent visual language and behaviour throughout the appsin the system. The aim is to eradicate the endemic problemspreviously mentioned. Current systems that I have analysed allsuffer from dysfunctional elements that are caused by a lack ofarchitecture. I propose much better can be done by providing aconceptual framework on the basis of which these apps areconstructed from.

As a student who spent majority of his life dealing with digitalsystems and have seen others go through the same difficulties, itfelt worthwhile to come up with a solution to this problem. Ipropose a solution based on my conviction that designing anenvironment with a unified user interface and consistent knowledgerepresentation throughout, would enable the student to stay on topof their academics, stay connected with their peers and teachers,and to a certain extend, help inform where to spend their limitedtime and focus budget on.

By following time proven software development techniques likemodular architecture and single responsibility principle, I arguethat it also simplifies management fromdevelopment perspective.

1.2 Acknowledgements

The ideas presented in this dissertation is a synthesis of variousnew approaches that have been proposed over the years in softwaremanagement and other fields of research.

I drew my inspirations from the teachings of great many people whichinclude my tutors, colleagues and friends. Many specific ideas haveemerged from discussions with many informal conversations I had withpeople some of whose names have escaped me.

My views about software development has been greatly transformed afterby Alan Kay, Alan Perils and Rich Hickey whose work and their principles onsoftware development. It helped me form the cornerstones in shaping my approachtowards creating software. I am deeply indebted to theClojure community for producing quality software and instilling in mesome of the best philosophies produced in the software community. Amore comprehensive list of all the technologies used can be found inthe appendix.

My ideas about metaphors and conceptual integrity come from GeorgeLakoff, Mark Johnson and Donald Norman.

I am grateful for the work of Edward Tufte's work on datavisualization which served as a source ofinspiration for the work in unifying visual interfaces.

TODO Decide name count.

For the evaluation phase, in what detail should I mention the names,all of them or just in passing?

All code developed towards creating this ecosystem has been opensourced and I look forward to see what comes of it.

1.3 Overview

In the following sections, I detail about various facets of theproject. Theory section describes meta ideas that formed conceptualgrounding for the dissertation. Readers would be able to see howthese abstract ideas were made concrete in method section. Furtherdetailing is provided in Discussion section where individual appsare put under scrutiny of the conceptual framework derived and theirinternal as well as overall consistency with these is analysed basedon the principles set forth.

Towards the end, I evaluate my dissertation based on the successcriteria stipulated, describe my take aways from the project and setbacks I faced. I also detail on what I assume a potential future ofthis project might be if further work is to be carried out.

I argue that by following the single responsibility principle, a lotof these problems can be eradicated and bring much more clarity froma user's perspective.

None of them is sufficient on it's own to give us a complete,consistent, and comprehensive understanding of all these aspects,but together they do the job of giving us a coherent understandingof what a rational argument is.

the amount of things the user has to think about when in thecontext of manipulating a single domain where as in something likethe Blackboard, a user is bombarded with a lot of questions about alot of domains and how do is?

2.1 The Two Worlds

This dissertation has been in major parts been one of building twodifferent worlds: The user facing interface of the system andthe underlying computations.

This is a visualization of how I model the two systems: The upperlayer being the front end user facing side of the underlyingstructures. This division is apparent in [Do the TVTropes research]and can also be seen in the Cyc product.

The dissertation will talk about how the ideas and concepts of thefirst world are more cogent than the logic layer that lays hiddenbeneath. It is also reflective of where my strengths and weaknesseslay.

These systems are built on a substrate of complex underlying systemswhere things are very fragile if you are not careful.

This is because the underground is always changing and couldpotentially lead to breaking down of whole software ecosystem ifthey are ever allowed to unmonitored changes. Though the softwaresystem presents many ways to organize this complexity, the basicnature of breaking on update is still incumbent/popular. [Invoke analogywith garden and roots that run the system. Cut the roots and thegarden disappears]. This is because the frameworks are ever changingowing to the whims of the makers. One way is to start from scratchand supply as much care and control

over everything by building themfrom the ground up but the scope of this dissertation as it had tobe produced in a limited time didn't allow for such levels ofexplorations.

H.G Wells in his famous book Time Machine invokes the image of aworld which is split into two. And it is said that what you doesn'tknow eats you and I had to experience this first hand.Pioneers of the user interface in Paolo Alto, called it the userillusion. Include reference. But in actuality the world

Keyword: Interoperability between technologies.

With a potential to step on each other's toes at least as often asthey stand on each other's shoulders.

The degree to which all the disparate pieces can work together.

We think in terms of metaphors, and the metaphors we use influence howwe think.

The gestalt of the system

The systems that I am about to describe can be understood in terms ofmetaphors relating to other systems. What I have envisioned is tocreate a system with a common system, as the term gestalt suggests isto create something that is more fundamental than the sum of it'sparts. The way I hope to achieve it is by creating a system which isseamlessly navigatable and employes a common lexicon of terms todescribe the various concepts that occur throughout the system.

Metaphors employed here are primarily deviced as to give a narrativestructure and give a fully coherent meaning towards what I was tryingto achieve in the end. It would be able to infer from thesepropositions how my actions were directed towards certain ends andwhenever certain things were made to be consistent in a particularway, I'll be invoking these metaphors to explain why certain tradeoffshad to be made for consistency.

On juggling mental models

As software construction goes it requires the frameworks to beinternally consistent and follow some contracts. Much like how afunction deals with an input and chains it on to another. But when adeveloper starts working with this, he is faced with the challenge ofkeeping everything intact.

Hell are other people's abstractions.

Picking libraries became an exercise in building taste on what ispalatable and if you identify with the author\342\200\231s ideologies.

On the dangers of metaphors

Leslie Lamport has an incisive essay on why the metaphor of biology islimiting in the sense that it gives an amorphous definition toprograms that is inherently complex and not understandable.

And in this sense verification is indeed needed but in my research itbecame quite an experimental field right now to adopt. [Providesupport here.] But given a chance and with enough knowledge next time,I would like to develop the systems in a formally verified way ratherthan using TDD and is something worth looking into if the system is tobe evolved into the future.

Systems that build systems

Using Clojure has disabused me of the praise people sing about it. Itis simply not so for an incoming newbie. You have to be well versed ona large pool of concepts to help it to see the programming domain inthe way it sees. So Clojure makes everything simple only after you arewell versed in Clojure or Lisp domain for that matter otherwise itonly adds to the noise.

Incidental Complexity

Controlling complexity has been one of the pivotal concerns insuccessfully establishing this project.

Building an Experential Gestalt

In their book, Metaphors We Live By, George Lakoff and Mark Johnsontalk of something known as experential gestalt. They contend that acluster of other components are experienced as a whole by human beingswhich they find more basic than the parts. My aim is to give theecosystem that I'm building a feel with all the individual appscollaborating in a seamless manner to form a coherent whole. Thisdirectly references the work of a prototype done by Rosch in her workon categorizations done by humans. A recurrent complex of properties,our concept of causation is at once holistic, analyzable into thoseproperties and capable of a wide range of variation. So one app in theecosystem can be seen as a prototype.

Fitting the right model

In management science the top down hierarchy or those trees that youhave been seeing through this text is mostly outperformed by crossfunctional teams. The key thing that has to be identified here is that

Questions to Self

??? Does this idea of interacting agents from AI add any value here?Can I think of my programs as interacting agents?

??? Does adopting FRP later in the line change the metaphor I have employed here?

Why So Meta?

Once I have enacted the metaphor, all the decisions and constraintsthat are imposed upon the system fall out from it. Since computer isan ultimate protean system, it allows us to model the environment inwhatever manner we please. It is the ultimate imitation medium. Theuser facing level is a universe and the underlying subsystems can beconsidered as a rhizome that originates from a central point and thenspreads out. These when represented out

References

http://www.ribbonfarm.com/2015/03/04/gardens-need-walls-on-boundaries-ritual-and-beauty/

2.2 Developing the Visual Language

The main challenges of developing the system is it's visuals. Aharmonious user interface with all the elements fitting each other.

system was to use a cogent visual language that covered the wholespectrum of the apps. A lot of principles in constructing the systemswere borrowed from Design of Everyday Things and the Tufte tetrology.

I greatly believe in the concept of personal computing as an amplifierfor human reach. And most if not all of the concepts governing the UIhas starts flowing having this as the central precept. Inorder tomaintain this, the things that had to be done were tight feedbackloop.

Conceptual Integrity was maintained. Designing with the user in mind.

Coherence over Consistency

Consistency leads to shoehorning the concepts whereas coherence givesan overall picture. Where possible consistency has been brought in,but I felt it's better to leave the consistency bit untouched becauseit requires a mature understanding of the ecosystem after it isunrolled and the user behaviours for the observation for about 6months inorder to see what's working and what's not. Only apreliminary understanding of this has been achieved so far and it wasmost sensible to leave it open. This allows new design languages to beincorporated into the system at a later stage.

Not an end all design philosophy

As mentioned previously. Things has to be in flux if evolving userneeds are to be satisfied. This means that the design guidelinesare to be evolved along as new apps are designed and never takethem to be steadfast rules but guidelines on getting the feel forthe app. Even without reuse of any elements, I believe thatsomething that honours the consistency within the system can bemade since there's more than just visual consistency that meets theeye that keeps the system coherent. This is discussed in thefollowing section.

The Philosophy

The philosophy we follow for the design is nothing new, rather it's asynthesis of ideas that has been floating around. We have tried tounify the shaded look that is characteristic of the skeumorphic eraand the flat design era and appropriate them to the right context. Togive a concrete example of how this plays out, check the followingsection:

Human Hours lost in Clicks

Caution has to be administered

One other thing that we have followed throughout is to minimize therequirement of user interaction with the elements on screen. Eventhought two clicks might only amount to 0.5 seconds. This two secondswhen conducted by 4000 users doing it 10 times means that it's 4000 *2 * 0.5 = 5 human hours on a big scale lost due to a dubious mistakefrom our side. Hence, we have supplied much of our attention inshowing the data and hiding away the irrelevant. This is not entirelyperfect yet, but we feel we have strived to adhere to this principlewherever possible.

The corollary of this is that actions that which are not usedfrequently are hidden behind two level actions.

Future

References

http://nxhx.org/RedefiningSoftware/

The case against user interface consistency
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.6480&rep=rep1&type=pdf

2.3 A Live Ecosystem

The ecosystem would not be complete without a supporting documentationof all the functionalities. This is implemented using a Clojure app.

The app is a blogging platform where developers can log in andauthenticate with the systems, obtain their license keys etc.

The documentation website contains 2 main sections, one for the user,one for the developer. The user section would be the front facingwebsite that would be available for all the UI materials that are onthe front facing website. Everything else is available on the websitewhen a developer logs in.

The ecosystem as it stands now can be recognized as a set of coresoftware and supporting structures around it. Lets go through thegeography and a little bit of history that necessitated the creationof these structures.

Core 7 apps and this landscape is constructed so as to allow as manycomponents as can be added to this system. This is enabled with thehelp of defining consistent interfaces that enables them to talk witheach other. One of the central precepts of this dissertation is thatof modularization and all these apps can be considered as an app thatconsists of even more modules inside them. Turtles all the way down.

Interconnected subsystems.

Design Process

Once systems are divided into relatively independent parts, the thirdstep of the design process occurs: creating visual representations asfit solutions for each sub-part. These sub-parts are then visuallyfitted together into a whole.

2.4 Software Development

Stories

Invoke Fred Brook's tarpit.

Architecture

Both the process and the product of planning, designing andconstructing buildings. Develop a language as you describe systemsbased on this architecture. This makes the architecture tangible andprovides an unambiguous description of architectural building blocksas well as concrete system while staying away from technologydecisions.

The architecture requires definition of the building blocks (types ofthings) from which the actual system will be bulit. It can be thoughtof as a language to talk about the system.

Giving something a name

Most interesting systems are complex because of the simple fact thatthere are a lot of entities and myriads of relationships amongthem. The only way we can hope to bring order to this chaos is byspotting patterns that helps us manipulate and reason about theseconcepts. As someone said: Giving a thing a name gives you controlover it.

Metaphors as Lenses

A note on the pictures. The pictures that you view in this are partialrepresentation. Think of them as metaphors to understand the realconcepts. Like a metaphor, they conceal some facets of the idea whilerevealing others. At places where apt, I have tried to view the sameidea from two different metaphorical lenses.

Hairball Rhizome

There are two ways to create complex stuff.Simple enough that there\342\200\231s no deficiency.Complex enough that there\342\200\231s no apparent deficiency.

Apart from that of an ecosystem another metaphor that can be used tosee this project is as that of a rhizome a thread of connection thatruns across it.

Silos and Snowclones

Components

Holistic vs. Reductionistic Approach

I have increasingly noticed that bespoke constructs and structures andperformance optimizations most of the time leads tocompromise in the generality of the software. This is not the mainfocal points of this dissertation. This is not to mean that they arede-emphasized but because of the nature of the project where the endusers requirements are not available prior to development, it is mostsensible to keep them flexible and build composable abstractions thatare as general as possible and helps to solve a wide range ofproblems. Focusing or improving upon a certain facet of a softwaremeans that if that component gets un-used in the end, all the effortgoes waste.

Software eats itself.

Better abstractions enable structuring your programs better.

WIP: Evaluation of Models

Lack of my knowledge with respect to software architectures in generaland inability to visualize the future of these apps have rendered tothis to be sub-optimal but the good thing about software is thateverything can be rebuilt from scratch once you have explored thedomain space thoroughly.

A few structures that I have in mind to architecture the ecosystem wasSupervisor-Workers model from Erlang OTP documentation.Attention had to be paid to the behaviour of this software.

Pros

Things are open. This level of transparency lets the designers tocreate their own systems and include the necessary changes in theother apps to reflect this. This means that interoperable technologies and othersystems that fit into the ecosystem can be introduced with much ease(Invoke the metaphor of LEGO here?)

The network intelligence does not reside in a single point ofcontrol.

Cons

References

Early History of Smalltalk

Erlang Documentation

http://www.erlang.org/doc/design_principles/des_princ.html

Systemantics

http://www.amazon.com/Systemantics-Systems-Work-Especially-They/dp/0812906748

On Visual Explanations

The dissertation employees a lot of visual explanations to make senseof the systems and explain the concepts.

Tree diagrams

Tree diagrams are basic structures with nodes and leaves. These areprimarly exercisesed in areas where the layout of the components ofthe system gave more sense to the work.

Metanarratives

Employed a technique called Literate Programming to explain all the code.This has been an exercise in improving the clarity of the intend to communicate it to myself and to others.

References

Learn about Processes, Alfred Whitehead and other things from:http://www.jfsowa.com/ontology/index.htm

Growing a Language

http://www.cs.virginia.edu/~evans/cs655/readings/steele.pdf

2.5 Growing the Ecosystem

Software doesn't exist in a vacuum, it is made alive with the peopleusing the software to meet their ends. In this sense, it is offoremost importance to value their ends. This is what much of humancentered design argues for. Insert reference. The philosophy chosentowards this problem is to tackle it from the viewpoint of a humancentered process to foster a community around the systems. My attemptat doing so and the results are summarized in the analysis page.

People are central to any software architecture.

The ecosystem would not be complete without a supporting documentationof all the functionalities. This is done in Literate programming stylein websites generated with Clojure.

The documentation website contains 2 main sections, one for the user,one for the developer. The user section would be the front facingwebsite that would be available for all the UI materials that are onthe front facing website. Everything else is available on the websitewhen a developer logs in.

The ecosystem as it stands now can be recognized as a set of coresoftware and supporting structures around it. Lets go through thegeography and a little bit of history that necessitated the creationof these structures.

Core 7 apps and this landscape is constructed so as to allow as manycomponents as can be added to this system. This is enabled with thehelp of defining consistent interfaces that enables them to talk witheach other. One of the central precepts of this dissertation is thatof modularization and all these apps can be considered as an app thatconsists of even more modules inside them. Turtles all the way down.

Interconnected subsystems.

Design Process

Once systems are divided into relatively independent parts, the thirdstep of the design process occurs: creating visual representations asfit solutions for each sub-part. These sub-parts are then visuallyfitted together into a whole.

Supporting Websites

I have devoted a about 1/5th of my time developing this dissertation towards developing thesesupport structures since they were a vital aspect for the students tolearn about the new technologies in the ecosystem. One of thedrawbacks of spreading things around is that there a lot of pointsthat are not in the picture. It can be argued that a monolithic appwould have one single point and all the elements can be found out ifyou explore the same app. But when spreading the apps around. This isdone in the current ecosystem by consistently crosslinking the picturewith the. The dashboard mirrors the concept of a central platform andthen reaching out from it.

References

3.1 Analysis of Current Systems

3.2 On Research Methodology

The methodology followed here is to collect information from varioussources. Alan Kay speaks about thinking the present reality as aconstruct which is a result of amalgamation of different theories thatpeople have forged up and to reconstruct it, look beyond what is andcherry pick the set of consistent ideas that will help you form aharmonious approach.

Following this approach, I have tried to pick only the elements thatmatters constantly pruning these resources as needed and re-adjustingmy hypothesis in the light of new data.

Caveats

Confirmation Bias. You look where you steer. This has been somethingthat I am not totally sure that I have gotten rid of. If I am to standback and evaluate what I have done so far in the way I have doneit. 80% is because I have chosen the routes and the routes thendetermined where I will reach. Choosing the path is the hardest andthe funny thing I have learnt is that most of the times you will bewrong. But the beauty then again is that life even though feels shortgives you ample time to backtrack on your ideas. Especially becausethis is a domain where in my opinion, humans have the least clue onhow to create process that work and to write ones that work, let alone how a coordination of them.

I do I understand

Experential knowledge as the guidelight.

http://blog.ncase.me/i-do-and-i-understand/

3.3 Implementation

Premature optimization is the root of all evil.

Late binded partly because that fits with the style of my working andparty because this enables more things to be grasped.

On work

As with most new ideas, it originally happened in isolated fits and starts.

We would only know where we are going after we reach there.

Looking back with a rear view mirror.

Optimizing after the fact.

Premature optimization is the root of all evil.

How things can be made more abstract when boredom strikes.

Table of Contents

1 4.1 The Problem

State the problem each app is trying to solve.

State when it's a feature.

A set of features are not going to solve the problem.

Focus on the tradeoffs in evaluation.

Table of Contents

- 1. 4.2 Background

1 4.2 Background

The web consists of a platform.

The predominant architecture being one of client and server.

5.1 Evaluation

Computer technology changes quickly and as a result specific technicalknowledge, though useful today becomes outdated quickly. Seen fromthis lens the following were the things that this dissertationprovided me of value.

Time Log

Include in Appendix.

Path Followed visualization

What this project endowed me with

The decision to use functional programming has been great toappreciate the fact that there is more than one way to reach yourdestination. And as Alan Kay writes in Computer Software inScientific American, the intellectual limits of Computers are notyet understood and this has been an experience which more emphasizedthe need to understand clay so as to construct betterpots. Computers are capable of self-interpretation whose upperceiling has not yet been understood.

Growing a community

Every complex system is made alive not by the code but by theaction of people who use the system to their ends. Starts from March1 with Bundles and ModuleBridge. The toughest test of the system iswhen it tries to be useful in contexts where the designer didn'tindent it to work. It hence was an interesting experiment to see whatkind of uses people liked to put these systems into.

The Right Kind of Pessimism

The right kind of pessimism is one where you assess the problems evenafter something works. It is good to see these systems as good lookingor may be even well performing but one of the core values underlyingin the construction of such a system is to focus on the kinds ofthings that this disables and a few such views. But this is reallylimited as it's the creator evaluating these systems and it is any waygoing to be an attempt in futility because I'm behind the creation andthere's only so much that I can see. I will leave it to the reader toassess the system, break it down and take away the components if anyfrom the source code (if you can make your way through those lispparanthesis first)

Time Tracking

While tackling the project it is easy to miss the forest for thetrees and the trees for the forest but it is even tougher to miss theincumbent drought that is about to hit the forest. I have exercisedmy utmost efforts at crafting the apps here, but for someone at mylevel of expertise can\342\200\231t possible handle the extended usage andadaption of the ecosystem, constant pruning of the systems andsubsystems is to be maintained if the ecosystem is to be prolonged.

Future

If the use of the system is to be continued it has to be ensuredthat high precision performance tuning is only done after properlyunderstanding the problem domain. Even if all the unit testspasses it doesn't guarantee that some of the combination of thefunctions and certain characterists which can never be unit tested. Itis unreasonable to unit test everything. New changes introduce newpathways to failure.

These new forms of failure are difficult to see before the fact; attention is paid mostly tothe putative beneficial characteristics of the changes. Because these new, high

5.2 Results

The results were mostly positive. And a lot of criticism on the waythings have been put together. But as a proof of concept idea andfoundational structure the results prove a much better foundation thanthe current one. I think it owes a large share to the design languagewhich played a unifying language. Reduce this down to questions tofill in after evaluation.

Room for focused improvement

A lot of room has now been generated to focus on the improvement.

6.1 Looking Back

Looking back at this project I think it has been a big journey tryingto figure out a lot of Clojure. Learning some pretty life changingconcetps from functional programming lanuages.

6.2 Future

What the future holds.

6.2.1 Next Apps

The current ecosystem could really benefit from

6.2.1 Design Improvements

6.2.1 Code abstraction improvements

6.2.1 Performance improvements

6.2.2 OTP Architecture

Soft real time system. Bring in Erlang.

6.2.3 Administrator/Tutor frontends

This project cannot be deemed to a complete system without having acomplementary tutor frontend. While the project didn't give me roomfor creating a administrator frontend. It was also out of the scopebecause of the given time frame.

6.2.4 Multiverses