ISR Institute for Software Research
University of California, Irvine

Green Calico - Reengineering Requirements for Sustainability for a Collaborative Drawing Tool

Christopher Arciniega
University of California, Irvine
carcinie@uci.edu

Birgit Penzenstadler
University of California, Irvine
bpenzens@uci.edu

June 2014
ISR Technical Report # UCI-ISR-14-2

Institute for Software Research
ICS2 221
University of California, Irvine
Irvine, CA 92697-3455
isr.uci.edu

University of California Irvine
Institute for Software Research
Christopher Arciniega, Birgit Penzenstadler
TechReport UCI-ISR-14-2 June 2014

Green CALICO

Reengineering Requirements for Sustainability for a Collaborative Drawing Tool

Table of Contents

Abstract

The purpose of this Software Requirements Specification document is to provide a description of Calico under a green sustainability premise. That is, a traditionally designed software is being adapted to be provide non-functional requirements that fulfill the five dimensions of sustainability described in *Infusing Green: Requirements Engineering for Green in and Through Software Systems [5]*. Thus these requirements seek to fulfill human sustainability, social sustainability, economic sustainability, environmental sustainability, and technical sustainability as explained in that paper in regards to Calico. These and other terms are described in the glossary. On the context level, this document covers the business case, domain model, stakeholders, and objectives of Calico. On the system level, this document covers system vision, usage model, specific requirements, and constraints. The meanings of these terms are described at the top of their respective sections. The intended audiences of this document are software engineers, requirements engineers, and research professionals interested in green IT. Those interested should also read upon *Developing a sustainability non-functional requirements framework* [3] and *Safety, security, now sustainability: the non-functional requirement for the 21st* [4]. Green Calico is a client-server based application that facilitates communication and software design among software engineers in a sustainable manner. If interested in a more in-depth look at Calico's actual features search for the document by Mangano's dissertation [2], which is the document on which this theoretical Green Calico is based upon. From here on, Green Calico will be referred to simply as Calico.

This report is the result of a research project carried out by Christopher Arciniega over the summer quarter 2014 under the supervision of Birgit Penzenstadler at UC Irvine.

Acknowledgement: We would like to thank Prof. André van der Hoek for feedback on an earlier version of this specification.

Business Case

A business case is an argument, usually documented, that is intended to convince a decision maker to approve some kind of action. [6] A Business Case includes an Executive Summary that describes the potential business being proposed. These summaries are, as the title indicates, intended for executives, or people who don't want to waste time reading a whole document before knowing what is in it. The Problem Statement goes in depth into what the business proposal is intending to solve, or in other words, how the product offers solutions, and therefore value for which people may be willing to pay. The Analysis then goes in depth into the problem. The Solution Options then illustrates the available solutions and how they fall short. The Project Description then offers the proposed product as the one of the alternatives. A Cost-Benefit analysis together with Recommendations proves how the product has a viable market through.

Executive Summary

Selecting between designing on a whiteboard and designing on digital interface just got easier when designers can design on both technologies at the same time. With Calico, there is no need for compromise, because you get the best of both worlds. Calico was designed with designers in mind, blending both major design platforms into one seamless package, because all our design energies should be focused on our actual designs, not on the pencil on which we draw upon.

Problem Statement

When it comes to software design, designers have the dilemma of choosing between designing on a whiteboard and designing on a digital interface. While designing on whiteboard provides flexibility, it can be overly simplistic and disorganized. But while designing on a digital interface can be "streamlined" and organized, it often times is too complex, cluttered, and even restrictive. Designers have to compromise between one or the other. Most designers choose to settle with the whiteboard [1].

Analysis

Unlike buildings or furniture, software occupies an abstract space with no inherent physical mapping, yet often times diagrams are the most natural way for software developers to communicate. [1] Software developers use diagrams to share, to explain, to manipulate, and to brainstorm among developers. [1] Sketching on whiteboards is a crucial design flow process where many key decisions are made regarding the software. [2]

Drawing on a whiteboard is the most natural way of making diagrams because of its flexibility and fluidity. [2] A designer can focus on designing without having to worry about notations or conventions imposed by software design tools. [2] But the biggest problem with the whiteboard is that it is a 'passive medium', offering no way to facilitate sketching. [2]

A whiteboard can get pretty messy and cluttered after so much sketching. It's impossible to remain organized because you can't tell what you will draw in the future, or how much space

some sketches might need. If the whiteboard became full, you would have to erase some drawings, some that maybe you didn't want to erase, or you might have needed in the future. You would have to take pictures of every drawing if you want to look back at a drawing, and if you want to add more details to the drawing, you would have to redraw the whole thing! How painful is that to the design process!? Research has shown that many times sketches done on a whiteboard are lost because it's too much of a hassle to digitize them. [1] The whiteboard seems to be the prevalent platform among designers, but when interviewed, many developers that have to work with whiteboards have expressed a desire for some kind of 'intelligent whiteboard'. [1]

The problem with digital interfaces is that although the software have unique features the features are on separate tools, and not comprehensive towards design behaviour: they may lack features. [1] Not only that, but many of the software impose design conventions, such as having you draw UML diagrams. [1] What if a designer prefers another expressive mode?

These setbacks from both platforms cause unnecessary economical back draws of resources in terms of time, money, and environmental impact. The more productive a designer can be the less resources will be consumed, the faster a product can be put on the market place. This is a matter of logic. There is always a demand for efficiency in business.

Solution Options

Designing should be an enjoyable experience with minimum frustration from the design platform. One option is using poster board drawings. The paper is large so one can draw large enough, and the diagrams would be saved. It would share the flexibility of a whiteboard, the only problem is that you cannot erase marker. This doesn't work so well.

Another solution to the whiteboard are the many electronic whiteboard software that has been made. The problem with these is that they are all "designed to support a particular way of working at a whiteboard." [2] This impedes the inherent ease with which one works at a whiteboard where you don't have to follow any particular design paradigm.

Designers don't want to compromise one benefit for another. Can a whiteboard and an electronic platform be combined in a natural way? The answer is yes. Our last solution is to make such platform that stays true to the whiteboard experience. And that is where Calico comes in.

Project Description

The vision of Calico is that using Calico should be as intuitive as using a whiteboard, and minimally streamlined as using a digital platform. Both technologies can mitigate each other's setbacks while building on each other's strengths. For example, sketching on a whiteboard is limited to the size of the whiteboard, but with the electronic components, a user would be able to

pan the canvas to never run out of space. Users would be able to use Calico collaboratively and independently. Because of it's minimalistic interface, sketching on Calico would be intuitive and there would be a very minimal learning curve.

To implement the software, one would need testing platforms. These include three Mac computers triple-booting with Windows and Linux as well as two android tablets, two iOS tablets, an electronic whiteboard, and a server. All these have to have access to power, and a network connection. The mac computers contain the three main operating systems in order to avoid purchasing more computers than are necessary. The tablets are minimized to two each because they are only needed for testing, and to save on hardware costs.

In terms of human resources, one need only four people to code up as well as design all the features of the software in a relatively quick time frame of about 12 weeks. One person would specialize in software design, one other would be a software engineer, and the other two would be programmers. It would be great if the engineer were knowledgeable in software protection and security as well. If not, one would have to hire a security engineer as well. The average salary of a software engineer based off of www.indeed.com is $90k, that of a security engineer is $88k, that of a software designer is $58k, and that of a programmer is $79k. One would also need a web designer to set up the software website and they average $68k per year. And last but not least an environmental engineer ,averaging $82k per year, would take a look at the environmental impacts along the way and make suggestions.

Developing the software would be divided into 4 phases. The first phase consists of developing for the software for the whiteboard, the android machine, and the windows machine. This would consist of the beta release, and this would take around 5 weeks. After the release phase 2 consists of extending to the Mac, which should take 3 weeks. Phase 3 consists of extending to the ipad, that consists of 2 weeks, then phase 4 consists of extending to the linux platform, which would consist of another 3 weeks. These milestones will serve as markers of progress. And operating systems are prioritized in order of popularity. We want to cover the most popular platforms first. We also want to release the software to these markets in an on going basis, rather than as an all out release. This is in order to secure the larger markets first. As software is released the engineer and one programmer would continue working of the software at hand while the other programmer would stay back and fix bugs from previous releases.

A major assumption here is that there would be no major bugs or design flaws in the software that would set back the project a substantial amount of time and resources. Were that to happen, the build time would have to be extended, and some employees would be dropped, such as the environmental engineer, the software designer, and the web designer. The work of latter two is most substantial during initial development, and the work of the environmental engineer is only supplementary, and not required.

At the end development we would also need to hire copyright attorney to review the software licenses for all components of the software, which would be drafted by the software engineer.

Such an attorney would cost $200 per hour according to www.nolo.com, which is why we want the engineer to write the licenses first. The attorney would be hired for a total of 16 hours.

Now for calculating costs: Overall the cost of the hardware would be $1,800 for the mac minis, $800 for 3 monitors, $600 for two ipad minis, two android tablets for $300, $16,000 for a server, an electronic whiteboard for $2500. The salaries for 20 weeks would be $22,800 for each programmer, $25,800 for the software engineer, $25,200 for the security engineer, $16,800 for the software designer, $19,800 for the web designer, $23,400 for the environmental engineer, and $3,200 for the copyright attorney. I also estimate $14,000 for rent and energy expenses. This all totals to $192,600. I would bump the budget up to $200,000 to be safe, and this is already coming from a generous estimate that assumes all the merchandise is bought new, and that all these employees are needed.

Say Calico sold individual licenses for $20 (1 user), $100 for startup licenses (3 users), and $500 for commercial licenses (20 users). And say that 2000 individual licenses were sold, 1000 start up licenses were sold, and 500 commercial licenses were sold in a year. That would total to $390,000, and an ROI of .95 in the first year.

An ROI of 1 and higher would be a successful project as investment would have more than doubled. Anything above 0, but below 1 was an ok investment. In terms of the product itself, the product should completed within 15 weeks and be portable to the three major operating systems. Anything above 15 weeks and the project has been moving too slow, and cost would be significantly higher.

Cost-benefit Analysis

Paper

Pros	Con
● Very cheap ● Work is saved ● intuitive/natural	● Very limited space ● Non-transposable drawings ● Cluttered/bulky paper to deal with ● Must replace paper often ● Must replace markers ● Noisy

The Whiteboard

Pros	Cons
● Cheap ● Flexible ● Use your own notations ● Follow your own conventions ● Intuitive/natural	● Limited space ● Rudimentary ● Unorganized ● Diagrams can get lost ● No inherent way to save drawings

	● Drawings are not transposable ● Passive medium ● Must replace markers ● Board can look messy/hard to read

The Electronic Whiteboard

Pros	Cons
● Transposable drawings ● Saves work ● Enhances diagrams ● Work organized ● Unlimited space ● Many options	● Price tag (some) ● Imposed conventions ● Imposed notations ● Steep learning curve ● Cluttered ● Uses energy ● Unintuitive

Calico

Pros	Cons
● Unlimited space ● Uncluttered ● Intuitive/natural ● Flexible ● Use your own notations ● Follow your own conventions ● Real-time Collaboration regardless of distance ● Minimalistic	● Price tag ● Uses energy ● Requires more hardware

From the cost benefit analysis, it is clear that Calico poses the greatest benefit for the least amount of detriments. It is therefore imperative that Calico reach the market. The return on investment of Calico should happen well within a year up to two.

Recommendations

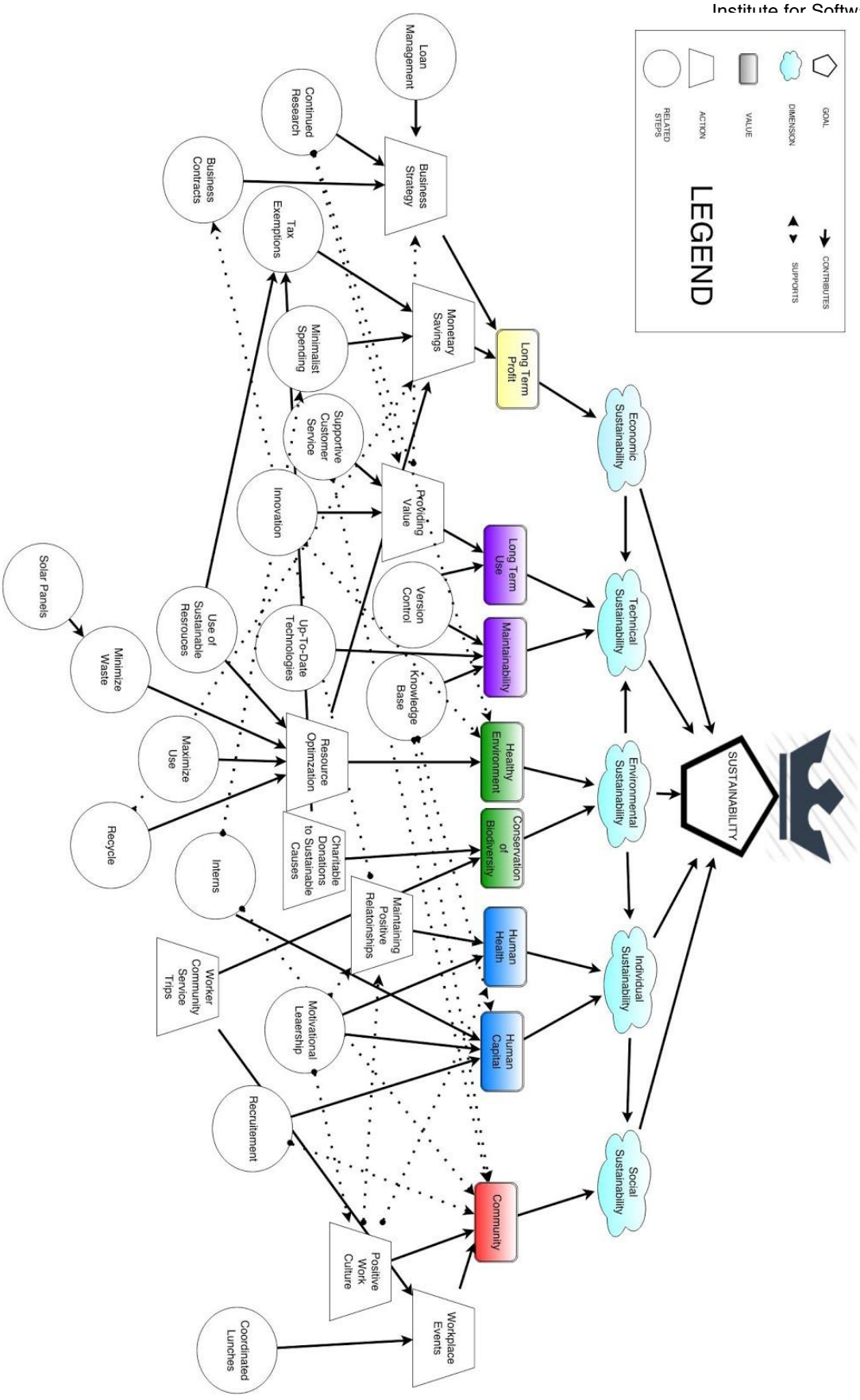
There is no software out there like Calico today that delivers all the benefits of a whiteboard and an electronic platform, without all the cons of either platform or newly created ones. Therefore no time should be wasted seizing this opportunity. Provided the research, and the demand for a platform that is actually better than the whiteboard demand for such a product would be overflowing. Be it that funds are required to run the project a venture capitalist should be sought in order to secure the $200,000 and his/her share should be limited to 80%. If only $100,000 are needed the profit should be reduced to 40%. Such ratio should be maintained.

Calico should be sold on varying licenses. One for individuals, another for start ups, and another for businesses. They should be priced $20, $100, and $500. The first two licenses are comparatively cheap to establish the market so businesses can gleam the popularity and success of Calico early on and be more likely to purchase the licenses. Calico should not be sold on a SaaS for the first two licenses because the way Calico is used, it is most heavily used in the beginning of design, and selling the software as a service would discourage purchase since it might not even be used for a prolonged period of time. From its uses Calico can be sold to software design companies on a 5-year license. Sales should also push Calico's use from the software domain to other design domains and open new markets. Since it could also be used for other collaborative design such as the design of hardware, appliances, etc. During development, a reevaluation of the assets and progress should be undertaken. At some point after launch and substantial profits have been made (ROI .90+) , management would be established for sales, engineering, quality assurance, and research so Calico can be maintained alive competitively for a prolonged period of time.

Sustainability Plan

Adapting non-functional requirements that fulfill the five dimensions of sustainability described in *Infusing Green: Requirements Engineering for Green in and Through Software Systems [5]*. The requirements seek to fulfill human sustainability, social sustainability, economic sustainability, environmental sustainability, and technical sustainability. The benefit of adopting such a goal is in the long-term health of the people involved, the business, and the environment. Being a long-term plan, the company is not expected to implement these within the two years of starting up.

Below is a diagram summarizing and illustrating the plan. The plan can be read in full detail below the diagram.


LEGEND
GOAL
DIMENSION
VALUE
ACTION
RELATED STEPS
CONTRIBUTES
SUPPORTS
SUSTAINABILITY
Economic Sustainability
Technical Sustainability
Environmental Sustainability
Individual Sustainability
Social Sustainability
Long Term Profit
Long Term Use
Maintainability
Healthy Environment
Conservation of Biodiversity
Human Health
Human Capital
Community
Loan Management
Continued Research
Business Contracts
Business Strategy
Tax Exemptions
Monetary Savings
Minimalist Spending
Supportive Customer Service
Innovation
Providing Value
Version Control
Knowledge Base
Up-To-Date Technologies
Resource Optimization
Charitable Donations to Sustainable Causes
Maintaining Positive Relationships
Motivational Leaership
Worker Community Service Trips
Interns
Recruitement
Positive Work Culture
Workplace Events
Coordinated Lunches
Solar Panels
Use of Sustainable Resrouces
Minimize Waste
Maximize Use
Recycle

Goal – Actualizing sustainability within the five domains

Economic Sustainability

- Long Term Profit
    - Business Strategy: the plans that will keep the whole system going (providing value) competitively on the market.
        - Loan Management
            - Limiting loans to a maximum of 30% of the expenditure
            - Making timely payments
            - Taking loans with minimal interest
        - Continued Research
            - Maintaining an ongoing research that keeps the software relative to the people using it and the technology available
            - Research would also involve ways to make the platform more energy efficient in terms of hardware that expends less energy, algorithms that are faster, and business strategy that is more efficient
        - Business Contracts
            - Maintaining healthy, loyal relationships with businesses that use the software and businesses that contribute to the software (directly, or indirectly)
            - Reinforced by Maintaining Positive Relations discussed under Individual Health
        - Reinforced by Monetary Savings, discussed below
    - Monetary Savings – saving money, and curtailing spending goes a long way
        - Tax Exemptions – applying, and actively looking for tax exemptions reduces costs
            - Small business tax exemptions
            - Environmentally safe tax exemptions
            - Reinforced by use of sustainable resources and charitable donations
        - Minimalist Spending
            - Buying Used – instead of buying the latest, and greatest of everything, buying what works and is used to save money
            - Buying only what is necessary
            - Buying wholesale strategically
            - Encouraging minimal use of consumables by employees
                - Ex. Toilet paper, paper, ink, lighting
            - Reinforced by minimalist spending
        - Reinforced by recycling

Technical Sustainability

- Long Term Use
- Providing Value

    - Innovation
        - The business should push for innovation and ensure good experiences with the software so the company stays relative and up-to-date
        - Involves pushing technological limits
    - Supportive customer service
        - Customer service should be helpful, and take care of customers
            - Customer service should be on average evaluated 5/5 by customers
    - Knowledge Base
        - Technical knowledge to do tasks/etc should be passed down through a private CalicoWiki by engineers
        - The Wiki should be kept up to date
    - Version Control
        - Maintaining an on-going version control of the software will help development and long term use by users
    - Up-To-Date Technologies
        - Database software, libraries, and programming languages should be future oriented, meaning that the technologies are trending into the future
        - Old technology should be upgraded in an ongoing basis

Environmental Sustainability

- Healthy Environment
- Reinforced by research
    - Resource Optimization
- Use of Sustainable Resources
    - Workplace:
        - Use recyclable consumables
            - Aluminum cans versus plastic bottles
            - Wooden pencils versus plastic mechanical pencils
            - Aluminum pens versus plastic pens
            - Bamboo where applicable versus wood
        - Minimize plastic
            - In electronics, consumables, construction, tools, waste bags
    - Use solar panels
- Minimize Waste
    - Resources should be consumed until faulty, or completely used
        - Workplace
            - Toilet paper
            - Printing paper
            - Electronics
- Maximize use
    - Maximize efficiency of workers
        - Work with personal physiological clocks
            - Workers should take 15 minute breaks every two hours

        - Compensate workers for excellence
    - Recycle
        - Recycle plastic, wood, metal, paper, electronics, preferably for money
        - Re-Sell electronics when no longer needed
        - Conservation of Biodiversity
- Charitable Donations
    - Donations would be made to environmental sustainability causes
        - Tax reductions also
- Worker Community Service Trips
    - Weekend service trips for workers
        - Builds community also

Individual Sustainability

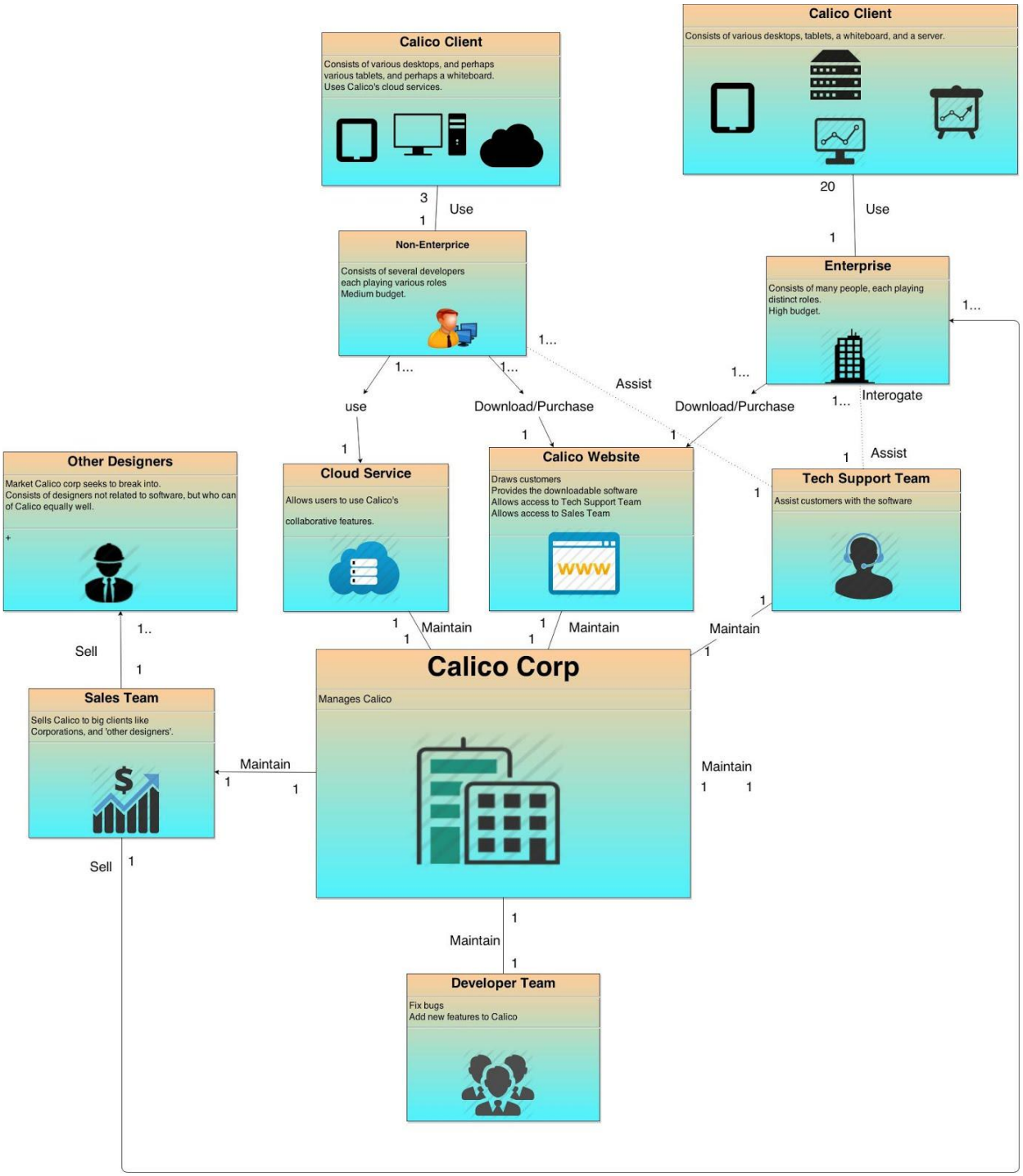
    - Human Health
- Motivational Leadership
    - Managers, and executives should give motivational speeches, and talk to their workers as if they are on the same level.
    - Leaders should pour their heart into their work and inspire that same passion in their workers.
    - Everyone should also recommend entrepreneurial books amongst themselves, and look after each other as a family
    - Workers hired work there for life, because they are family, if a worker is having a hard time they are counseled and helped.
        - If the company is having a hard time, instead of having individual people pay by getting laid off, the whole crew pays with variable unpaid vacations
    - Workers given free book “Atlas Shrugged” by Ayn Rand
- Maintaining Positive Relationships
    - Workers are encourages to know each other well
        - Lunches/events foster this
    - Sour relationships and dealings are frowned upon
        - Be it in the workplace or with business partners
    - Workers given free book “Nonviolent Communication” and encouraged to read it
    - Reinforced by Motivational Leadership and Positive Work Culture
    - Human Capital
- Develop unpaid interns
    - Diligent interns later become full-time workers
- Recruitment
    - Compensate workers for recruiting adept workers
    - Workers choose who they work with. They do the interviews.

Social Sustainability

    - Community
- Reinforced by Knowledge Base, Supportive Customer Service, Interns, and Recruitment
- Positive Work Culture

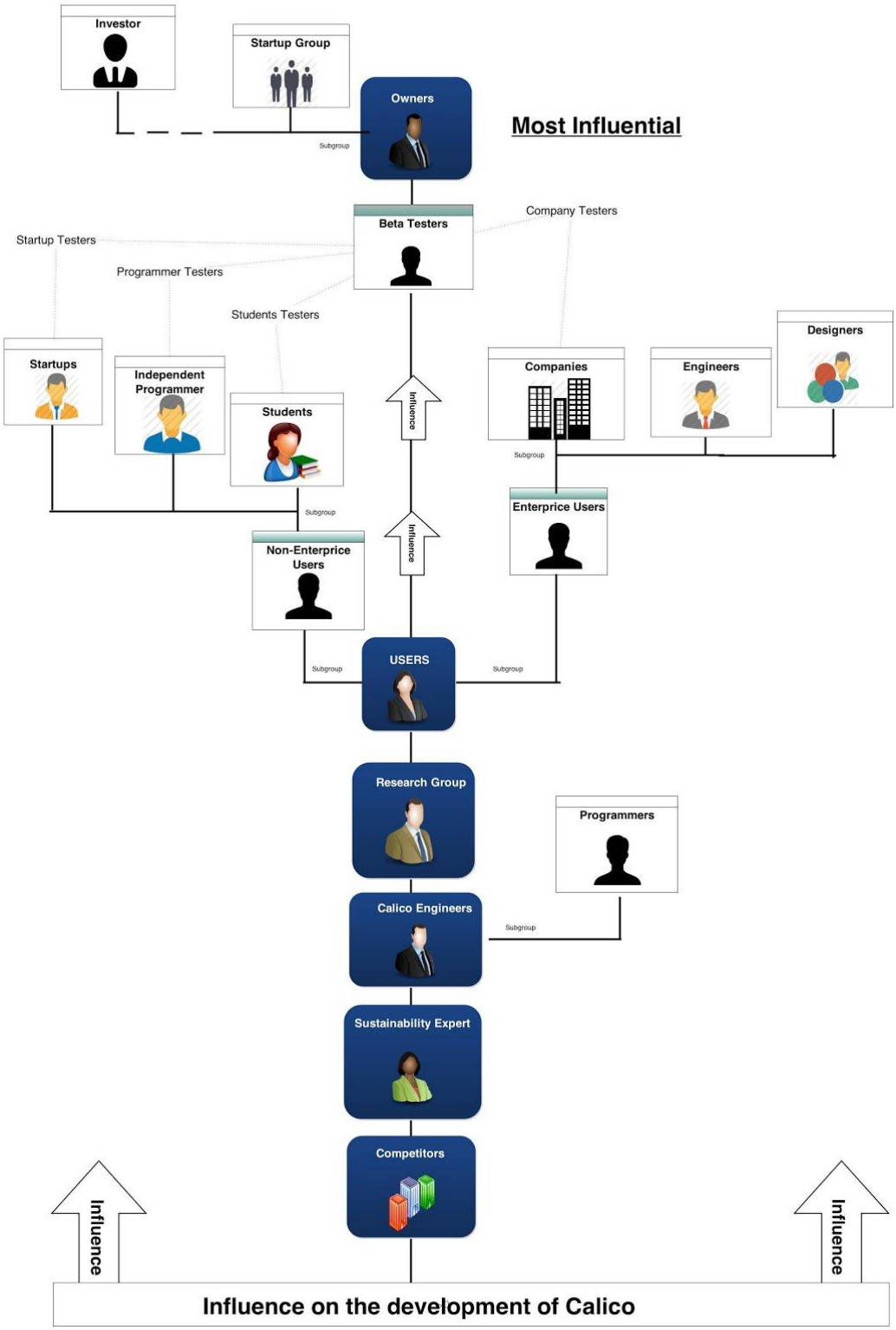
    - Reinforced by Motivational Leadership
- Workplace Events
    - Worker Community Service Trips
    - Coordinated lunches

Domain Models – "*A domain model in problem solving and software engineering is a conceptual model of all the topics related to a specific problem. It describes the various en66es, their attributes, roles, and relationships, plus the constraints that govern the problem domain.*" It "captures the most important types of objects in the context of the business. The domain model represents the 'things' that exist or events that transpire in the business environment." [6]



University of California Irvine
Institute for Software Research
Christopher Arciniega, Birgit Penzenstadler
TechReport UCI-ISR-14-2 June 2014

Stakeholders – *Stakeholders are entities that have influence over the process of developing Calico. "A stakeholder is a person or organization who influences a system's requirements or who is impacted by this system." [6] Interest may involve be positive or negative motives.*



Stakeholder Table

Investor	*Motivation: To get the greatest return with least amount of investment in Calico.* *Authority: Depends on the amount of power given by the Startup Group. Many times, more powerful than the Startup Group, if investment was large.* *Expectation: Want Calico to be successful. Want the company to grow as big as possible and control the largest share on the market.* *Expertise: Visionaries with an eye for successful companies.* *Availability: Only for crucial matters concerning the direction in which the company is going.* *Relation: Variably the highest position*
Startup Group	*Motivation: To get rich from Calico. Have the greatest desire to see Calico go to completion.* *Authority: All the power (unless given to investor)* *Expectation: Want Calico to be successful. Want the company to grow as big as possible and control the largest share on the market.* *Expertise: Formulated the vision of Calico.* *Availability: Only for very important matters* *Relation: Highest position*
Owners	*Decision makers*
Beta Testers	*Motivation: To shape Calico in the way that best suits their own individual needs.* *Authority: First to hop on board, and first users to have a say in early design phase of Calico, where more can be done.* *Surmount to the early gossip of Calico.* *If the early adopters really like Calico, Calico will have an* *easy time from all the rave. They will act like a springboard for launch, otherwise they will be a detriment where Calico can be left in the shadows.* *Expectation: Want Calico to fulfill all their design needs. Expect bugs. Expect to be heard. Expect new features upon software release.* *Expertise: Software design knowledge.*

	Availability: Highly available. *Relation: First users*
Designers	*Motivation: To design software that satisfies their employer* *Authority: Towards Company* *Expectation: To be able to communicate to large teams, work synchronously, in an efficient way.* *Expertise: Design* *Availability: Adequate* *Relation: End-User employee*
Engineers	*Motivation: Seek and easy way to communicate between engineers.* *Authority: Towards company* *Expectation: Requires that Calico be as intuitive and non-intrusive as possible* *Expertise: Engineer* *Availability: Adequate* *Relation: End-User employee*
Companies	*Motivation: Improve design efficiency and communication between employees.* *Authority: Priority end-user* *Expectation: Expects Calico corp to cater at a much higher level.* *Expertise: Multiple* *Availability: High* *Relation: End-user*
Startups	*Motivation: Highly motivated to design their own breakthrough software.* *Authority: Potential company that would make use of software in future. Pays more for software.* *Expectation: Needs Calico to be able to*

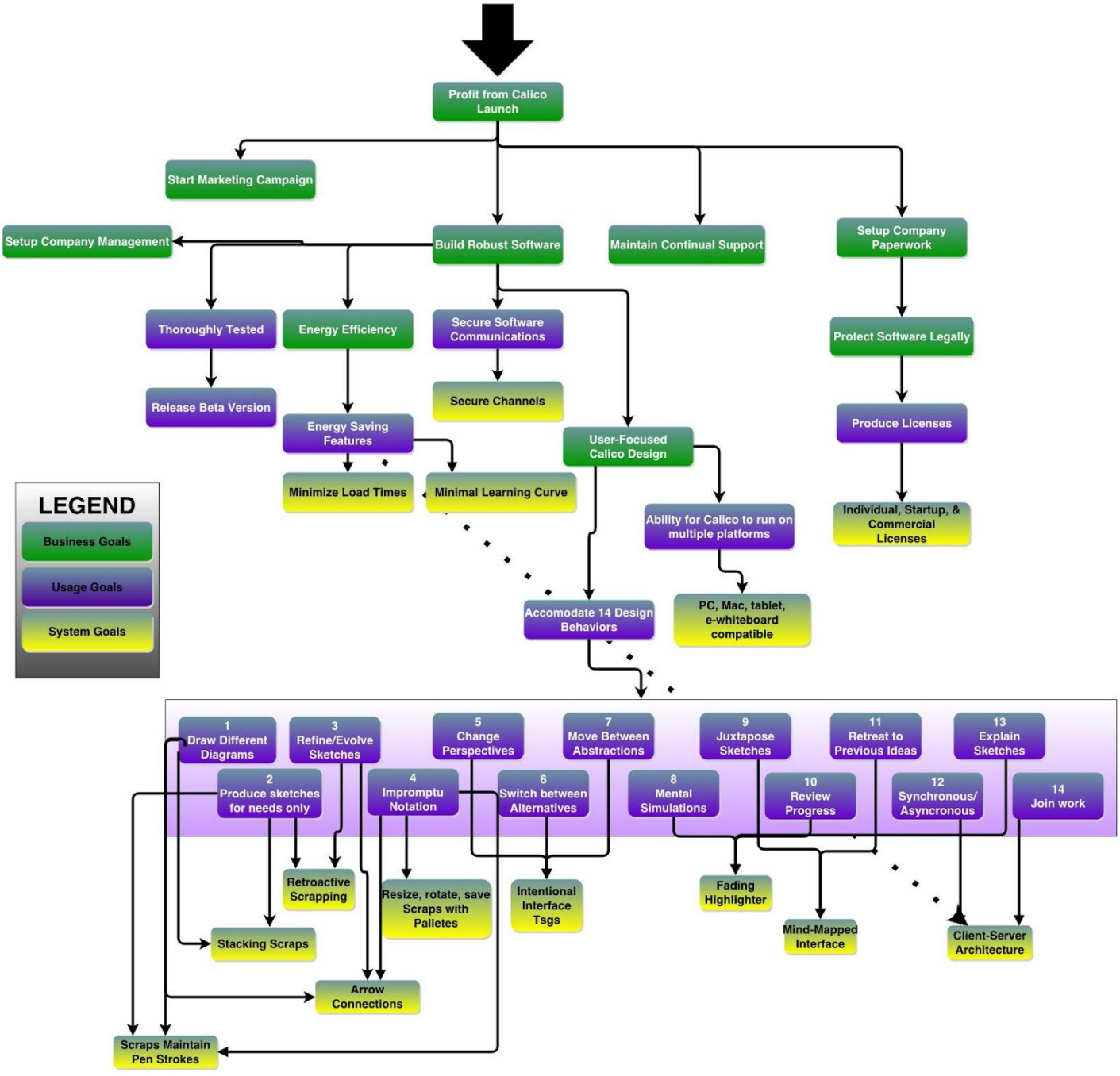
	organize ideas well, and help communicate between partners. *Expertise: Software* *Availability: Medium* *Relation: End-user*
Enterprise Users	*Motivation: Need the software on a regular basis.* *Authority: High-end user* *Expectation: Want higher ease of use because they are exposed to more of the software, and so become aware of more inefficiencies, and flaws. Also have a greater desire for Calico to cater to their needs.* *Expertise: Computer Science* *Availability: High* *Relation: End-user*
Independent Programmer	*Motivation: Might need software for keeping design organized. Some might design their own plugins for Calico.* *Authority: Low* *Expectation: That Calico be cheap for their limited use.* *Expertise: Programming/Designing* *Availability: Mediocre* *Relation: End-user*
Students	*Motivation: Might need software for class, or for own project.* *Authority: Low* *Expectation: Does not want to pay for Calico, or wants a student discount. Use of Calico might be very minimal.* *Expertise: Computer Science related degree in progress. Novice.*

	Availability: Low *Relation: End-user*
Non-Enterprice Users	*Motivation: Only need the software for a limited time.* *Authority: Least authority of all users, but high in numbers could gain substantial authority.* *Expectation: Want Calico to be free, or very cheap. Don't want Calico to impose certain notations or conventions. Desire good end-user support.* *Expertise: Design/Programming* *Availability: Moderate* *Relation: End-user*
Users	*Purchase the Calico software.* *Each user has their own idea about how to use Calico, or what features Calico should have.*
Research Group	*Motivation: To find ways in which Calico can better fit the natural tendencies and needs of software designers.* *Authority: Provide objective research-authority, with high probability of coming into play* *Expectation: Expect Calico to make use of their research* *Expertise: Observation* *Availability: High* *Relation: Constantly testing Calico and various methods in a social scale*

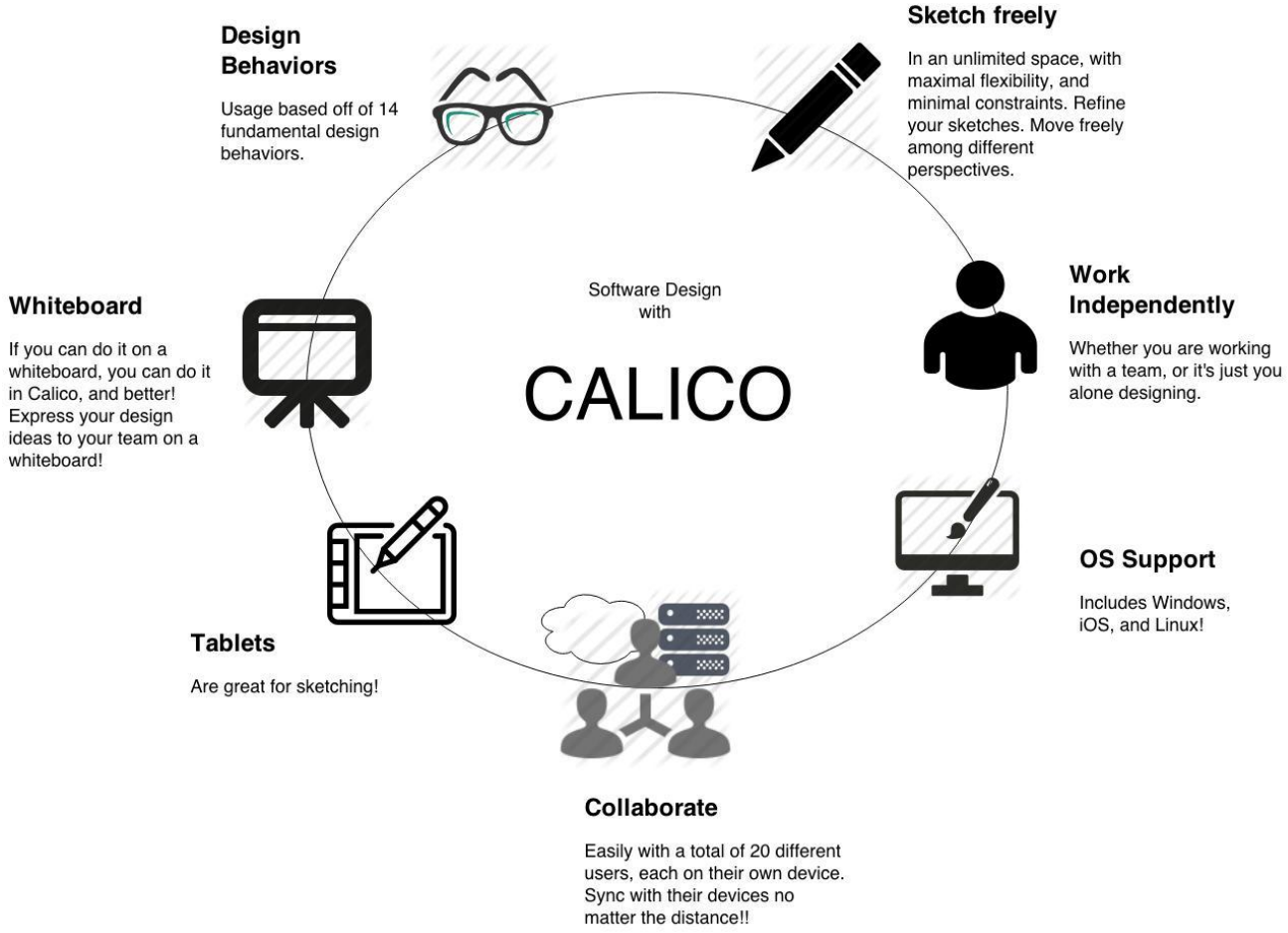
Programmers	*Motivation: To write the Calico software, and make it the best of its kind.* *Authority: Can make suggestions for big moves, but also design Calico at a detailed level.* *Expectation: Want the vision of Calico to be implementable. Seek management positions. Have their own ideas about what features or conventions Calico should have.* *Expertise: Writing code. Designing software.* *Availability: Busy* *Relation: Go-to people for actual software changes*
Calico Engineers	*Calico implementers*
Sustainability Expert	*Motivation: To reduce environmental impact.* *Authority: Power over sustainability matters. Must negotiate with management for costly moves.* *Expectation: Expects Calico management to follow advice over sustainability, even if it means greater cost for the company.* *Expertise: Environmentally Conscious* *Availability: High* *Relation: Environmentally has everyone's well-being in mind.*
Competitors	*Motivation: Seek to either acquire Calico, or destroy it. Desire Calico's clients.* *Authority: Authority based off of competition.* *Expectation: Expect Calico to behave competitively, and take away some clients.* *Expertise: Have their own version of their software that achieves the same goal, but perhaps in a different manner.*

	Availability: Not available. *Relation: Adversary of Calico.*

Objectives – *Illustrates the goals and reasoning behind the features of the software. "Goals form an essential part of the rationale for requirements and build a basis early on for traceable requirements, prioritization of requirements, [and so on]." [6]* Elicitation of the goals allows for one to illustrate why and how a system should be developed. [6]



System Vision – *Gives a birds eye view of the use and unique features of the software. "The system vision is a joint vision of the system agreed upon by all stakeholders." [6] It's purpose is to communicate to all stakeholders the agreed upon model, and it serves as the agreement.*



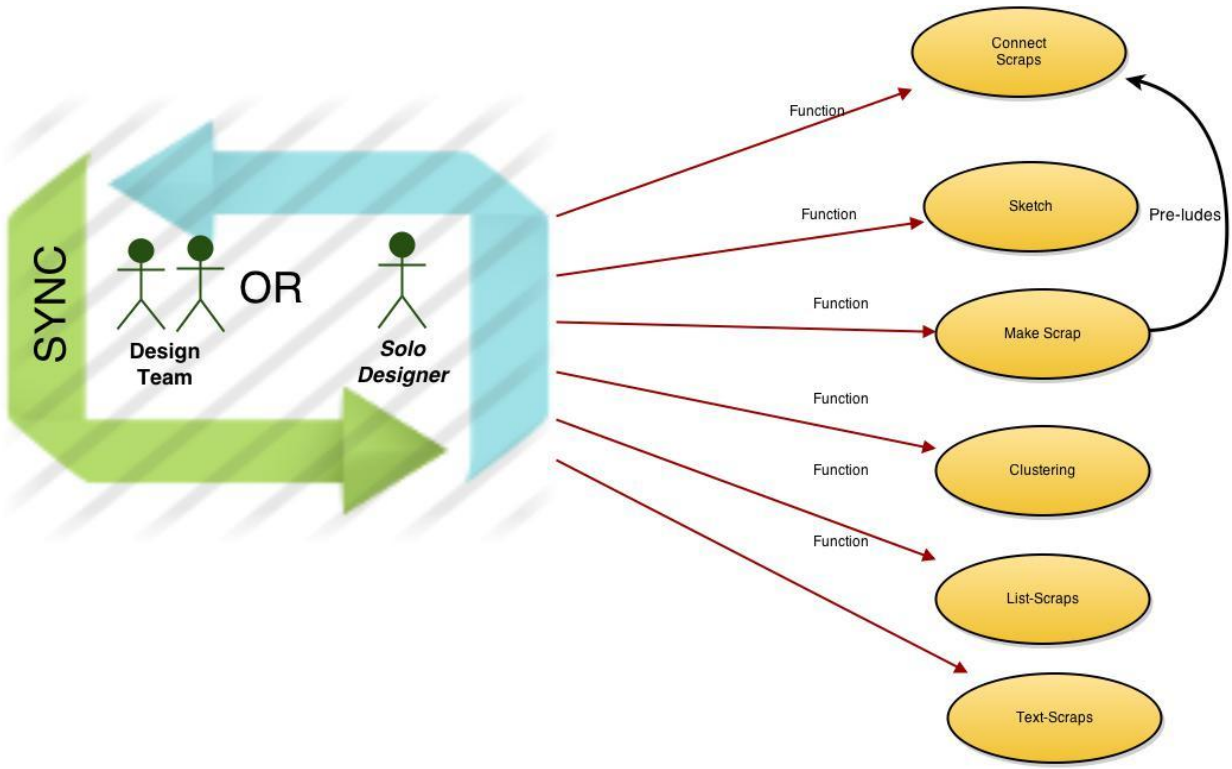
The software is to accommodate common design behaviors (14 total) efficient among the different design platforms (whiteboard, paper, digital)

· Collaborate - Designers should have the option to work on a sketch simultaneously, independent of location
· Must be able to refine sketches so that they can evolve over time
· Must be able to draw different kinds of diagrams on the same canvas
· Free-form notations – notation must unconstrained so it can be adaptable non-conventional, and improvised

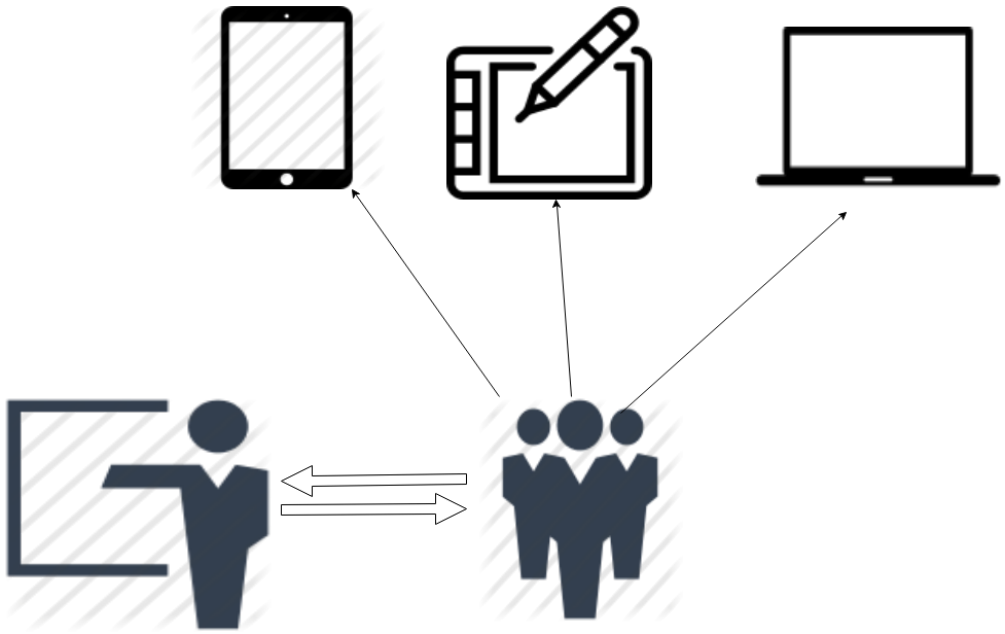
· Designers must be able to move readily from one perspective/abstraction to another with ease
    - One perspective may contain a totally different representation of the same instance
    - One level of abstraction can contain multiple sketches
· Must be able to juxtapose drawings
· Designers must be able to review their progress
· Designers must be able to retreat to previous design ideas and start over from there
· Must accommodate for switching between synchronous and asynchronous work

Usage Model – *Illustrates the various ways the software can be used. "A usage model describes the system behavior from the point of view of the user (Black-box) by modeling interaction sequences. It specifies the use cases. A use case is a series of system events triggered by an actor that leads to results for the actor." [6]*

Use Case Overview

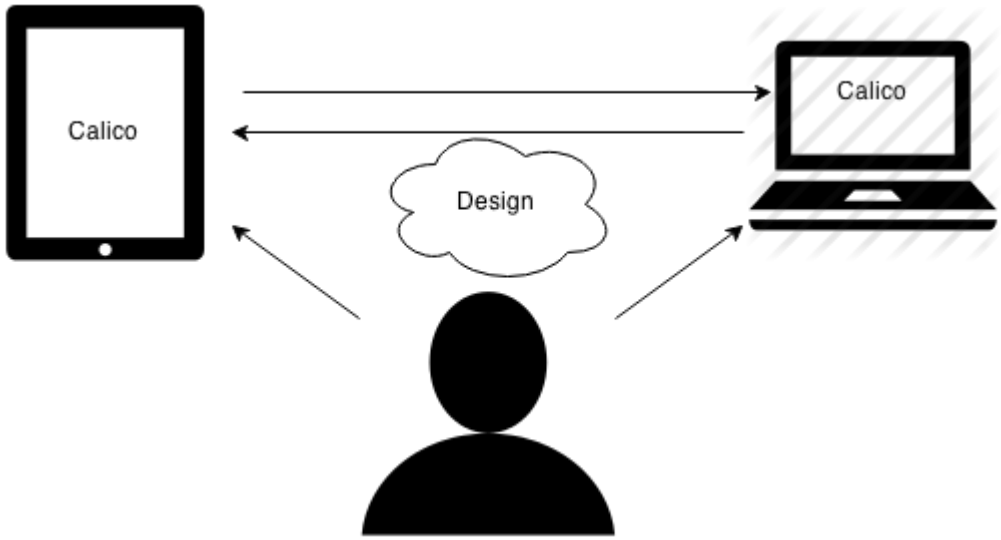


User Story 1: Collaborative Design



Various users, up to 20, use the Calico client on multiple tablets and computers, and one whiteboard. The group meets up to design a software. Perhaps many of the users developed ideas through Calico on their tablets, and laptops at home. Now they are assimilating their ideas into one. A group leader is standing next to the whiteboard and leading the design of the software. The team members ask questions, give advice, draw on their computers/tablets to show on the whiteboard, and exchange ideas. As the design progresses, perhaps the idea falls on its knees and the team reverts to another idea. After a session of design, the team members head back home to mull over the design. The next day, the team meets again and now the members come home with fresh ideas they developed where new conflicts arose as well as solutions. The team does not like the new design and they revert back to the old design. It is easy to see where they left off on their old design using Calico. From then on the team finishes the first design they had in mind. Now that the team is done designing, the team documents the design and presents it to the engineers that will be developing with the designers the software. A few of those designers were part of the engineering team. The engineering team easily assimilates the design seeing that it has been designed thoroughly using Calico.

User Story 2: Independent Design



An ambitious entrepreneurial Computer Science student has an idea and wants to develop it using Calico. He pulls out his tablet and starts designing on campus. After doing a few sketches he goes to class. After class the student goes home rushing to get on his computer which has a much bigger screen than his tablet. He pulls out calico and continues designing using a digital drawing pad. He sees his previous sketches and decides to add much more detail to his sketches. He then goes on to program the design.

- **Use Case:** 1 Connectivity
- CHARACTERISTIC INFORMATION
  - **Goal in Context**: Multiple teams, be it one or more people on each team, are able to connect to each other and work on the same project at the same time. Edits are always live. Asynchronous work is accomplished through working on different canvases at the same time. This is not interrupted by the live connectivity. Being constantly connected, synchronous work is easily achieved.
  - **Scope**: Client-Server Architecture
  - **Level**: Primary Task
  - **Preconditions**: The teams have the Calico clients installed on their respective platforms, and the server is running Calico.
  - **Success End Condition**: The platforms connect with the server (local or remote). Edits to the corresponding Calico project are live between the connected parties.
  - **Failed End Condition**: A number of platforms fail to connect to the server. Or connection is stale and edits aren't live.
  - **Primary Actor**: Design Team lead
  - **Trigger**: Start of design meeting between parties
- **MAIN SUCCESS SCENARIO**

1. Platforms are started
2. Calico is started
3. Platforms connect to their respective servers
4. Edits become live between connected parties

- **EXTENSIONS**

Single Individual Connection

1. Platform is started
2. Calico is started
3. Platform connects to the Calico cloud

Asynchronous Design Behaviour

5. Teams or individuals work on different canvases at the same time on their respective platforms

Synchronous Design Behaviour

5. Teams work on the same canvas at the same time on their respective platforms

- **RELATED INFORMATION**

– **Priority**: Critical
– **Performance Target**: Connection must be instant
– **Frequency**: At every use
– **Channel to primary actor**: Server (local or cloud)
– **Secondary Actors**: Team members be they separate, or grouped physically
– **Channel to Secondary Actors**: Server (local or cloud)

- **SCHEDULE**

– Due Date: Release date.

- **Use Case:** 2 Mind-Mapped Interface
- CHARACTERISTIC INFORMATION

– **Goal in Context**: The mind-mapped interface allows flexible control over the organization of the canvases which allows users to easily juxtapose sketches and jump around ideas. The user annotations are useful for labeling.
– **Scope**: Server connectivity
– **Level**: Primary Task
– **Preconditions**: There are multiple sketches in the Calico project.
– **Success End Condition**: The user is able to navigate between sketches with ease and is able to get a big picture sense of the work.
– **Failed End Condition**: The user is unable to navigate through the mind-map because of clutter or from too high a learning curve.
– **Primary Actor**: Design Team lead
– **Trigger**: Need for navigating through canvases

- **MAIN SUCCESS SCENARIO**

1. The user presses the button to go into the mind-map mode.
2. The user is able to see the drawings in an organized fashion

- **EXTENSIONS**

Quick Zoom

3. The user holds one finger on a sketch
4. The sketch enlarges to show detail

Root Navigation

3. The user clicks on a sketch three times
4. The view is switched so that the selected sketch becomes the root node

Extension:

5. The user goes back to the old root by pressing Root Home at the top

Labeling

3. The user selects a marker and begins sketching over the mind-map
4. The mind-map view accommodates those sketches when the view is enlarged or minimized

- **RELATED INFORMATION**
– **Priority**: Critical
– **Performance Target**: Sketches must be rendered to scale and instantly
– **Frequency**: In most uses
– **Channel to primary actor**: Calico Sketch Platform
- **SCHEDULE**
– Due Date: Release date.

• **Use Case:** 3 Scraps Maintain Pen Strokes

- **CHARACTERISTIC INFORMATION**
– **Goal in Context**: A user needs to draw on the canvas and the canvas needs to not interfere with the sketch. The strokes by the user must be represented accurately. Impromptu notation can be accomplished by these means as well.
– **Scope**: Drawing graphics
– **Level**: Primary Task
– **Preconditions**: A Calico canvas must be loaded
– **Success End Condition**: The user is able to draw, and the canvas represents the strokes
– **Failed End Condition**: The user cannot draw on the canvas, or the strokes are misrepresented
– **Primary Actor**: Designer
– **Trigger**: Moving pen or finger across canvas with a drawing tool selected
- **MAIN SUCCESS SCENARIO**

1. User clicks to see all the sketches in the workflow
2. Calico displays sketches
3. User loads a sketch
4. User selects a drawing tool
5. User begins moves pen or fingers across canvas
6. Canvas follows the users strokes with electronic ‘ink’

- **RELATED INFORMATION**
– **Priority**: Critical
– **Performance Target:** Strokes must be followed instantly

– **Frequency**: At every use
– **Channel to primary actor**: Either of computer, tablet, or whiteboard
– **Secondary Actors**: Any team members
– **Channel to Secondary Actors**: Server
• **SCHEDULE**
– **Due Date**: Date of release.

• **Use Case:** 4 Connected Scraps (Arrow Connections)

• **CHARACTERISTIC INFORMATION**
– **Goal in Context**: The user has the desire to show the relation among different scraps, so the user draws relations between scraps with arrows. With this the user is able to design different diagrams, refine, and evolve sketches through organization. Arrows also allow for impromptu notations.
– **Scope**: Scrap functionality
– **Level**: Secondary task
– **Preconditions**: There are multiple drawings on the canvas of which there are at least two scraps
– **Success End Condition**: Two scraps are connected with an arrow
– **Failed End Condition**: no arrow becomes present, or the arrow does not get attached to the scraps, or the user is never presented with the option to transform a stroke into a connector
– **Primary Actor**: Designer
– **Trigger**: The user drags the stylus from one scrap to another
• **MAIN SUCCESS SCENARIO**

1. The user drags the stylus from one scrap to another
2. The pen stroke becomes highlighted
3. An icon is presented to the user to transform the stroke into a connector
4. The user presses the icon and an arrow appears following the stroke
5. When the scraps move, the connector follows the two connected scraps accordingly

• **RELATED INFORMATION**
– **Priority**: Important feature
– **Performance Target:** The arrow should appear instantly, and move smoothly with the scraps
– **Frequency**: Moderate
– **Channel to primary actor:** Computer, tablet, or whiteboard
• **SCHEDULE**
– **Due Date**: Release date

• **Use Case:** 4 Retroactive Scrapping

• **CHARACTERISTIC INFORMATION**
– **Goal in Context**: A user wants to make a scrap out of objects drawn on a canvas. This means the user is able to control sketch production at will with no interference from automatic

computer intrusion. Thus this features allows users to produce sketches for needs only, and allows users to refine and evolve sketches.

– **Scope**: Scrap mechanism
– **Level**: Primary Task
– **Preconditions**: The Calico canvas has various sketches
– **Success End Condition**: The circumscribed sketches become a scrap that can be manipulated
– **Failed End Condition**: No scrap is formed from the objects, or only some of the objects are marked as a scrap, or every object is made into a single independent scrap
– **Primary Actor:** Designer
– **Trigger**: User presses the secondary button of stylus and drags their stylus

• **MAIN SUCCESS SCENARIO**

1. Calico has various sketches on the board
2. Using the secondary button on the stylus pen, the user circumscribes a drawing
3. Everything in the area circumscribed is highlighted with a thin border and grey background
4. The area is now a persistent object with a grey background
5. The user now drags the scrap into a palette to save the scrap
6. The scrap is now saved for later use and is viewable on the palette from any canvas

• **RELATED INFORMATION**
– **Priority**: Critical
– **Performance Target**: Scraps must be made instantly
– **Frequency**: Depending on usage, frequent, varied, or none at all
– **Channel to primary actor:** Either of computer, tablet, or whiteboard
• **SCHEDULE**
– **Due Date**: Release date.

• **Use Case:** 5 Explaining Sketches

• **CHARACTERISTIC INFORMATION**
– **Goal in Context**: User needs to point, circle, and draw sequences over diagrams temporarily in order to explain sketches to an audience, and to simulate over the sketches.
– **Scope**: Fading Highlighter mechanism
– **Level**: Primary Task
– **Preconditions**: The Calico canvas has various sketches
– **Success End Condition**: The user makes various highlights over the canvas, and those highlights fade to disappearance a few seconds after they are made
– **Failed End Condition**: The highlights are made permanent, or no highlight is made
– **Primary Actor:** Designer
-- **Secondary Actor:** Design Team
– **Trigger**: The user needs to explain a sketch or sketches to an audience

- **MAIN SUCCESS SCENARIO**
    1. The highlighter button is pressed
    2. The user begins to draw various highlighter strokes on the screen
    3. The fading highlighter creates a dark-golden stroke that is 3px wide
    4. The highlighter appears on the various other connected display machines
    5. The highlighter begins to disappear after 4 seconds until the stroke created with
       the fading highlighter is deleted.
- **RELATED INFORMATION**

– **Priority**: Critical
– **Performance Target:** Highlights must appear instantly on all screens
– **Frequency**: Frequent
– **Channel to primary actor:** Either of computer, tablet, or whiteboard
-- **Channel to secondary actor:** Either of computer, tablet, or whiteboard

- **SCHEDULE**

– **Due Date**: Release date.

- **Use Case:** 6 List-Scraps (Stacking-Scraps)
- **CHARACTERISTIC INFORMATION**

– **Goal in Context:** To organize scraps into a linear vertical list. This confers the user the freedom to organize and draw at will (design behaviour 1), as well as produce sketches for needs only (design behaviour 2).
– **Scope**: Scrap Mechanism
– **Level**: Secondary Task
– **Preconditions**: The Calico canvas has multiple canvases, and multiple scraps
– **Success End Condition:** Various scraps are aligned to a vertical list
– **Failed End Condition**: Clusters are not formed, or uncategorized
– **Primary Actor**: Designer
– **Trigger**: The user presses the text-scrap button on the menu-bar, or presses enter while in the canvas

- **MAIN SUCCESS SCENARIO**
    1. The user presses the text-scrap button
    2. A box appears
    3. The user drags scraps on top of the box
    4. The scraps are arranged automatically in vertical order with check-mark boxes, which can be checked, or unchecked
    5. The user then creates another list and drags that list to the first list
    6. The new list becomes nested in the old list
- **RELATED INFORMATION**

– **Priority**: Secondary
– **Performance Target:** List should appear and be organized instantly
– **Frequency**: Infrequent

– **Channel to primary actor:** Either of computer, tablet, or whiteboard
- **SCHEDULE**
– **Due Date:** Release date.

• **Use Case:** 7 Text-Scraps

- **CHARACTERISTIC INFORMATION**
– **Goal in Context**: Adding text scraps to canvas. This confers the user the freedom to organize and draw at will (design behaviour 1), as well as produce sketches for needs only (design behaviour 2).
– **Scope**: Scrap Mechanism
– **Level**: Primary Task
– **Preconditions**: The Calico client must be up and running
– **Success End Condition**: A text-scrap appears on the canvas
– **Failed End Condition**: No text-scrap appears, or the text-scrap that appears is not editable
– **Primary Actor**: Designer
– **Trigger**: The user presses the text-scrap button on the menu-bar, or presses enter while in the canvas
- **MAIN SUCCESS SCENARIO**
    1. The user presses the text-scrap button
    2. A box appears
    3. The types onto the scrap with the keyboard
- **RELATED INFORMATION**
– **Priority**: Secondary
– **Performance Target:** List should appear and be organized instantly
– **Frequency**: Frequent
– **Channel to primary actor**: Either of computer, tablet, or whiteboard
- **SCHEDULE**
– **Due Date**: Release date.

• **Use Case:** 8 Saving Scraps Through Palettes

- **CHARACTERISTIC INFORMATION**
– **Goal in Context**: The user must be able to save scraps in an easy to re-copy way. Pallets allow the user to stores scraps into the bordering interface for easy access between canvases. This features also helps with creating impromptu notation as notational diagrams can be reused with ease.
– **Scope**: Palette Mechanism
– **Level**: Secondary Task
– **Preconditions**: The Calico canvas must have a sketch of some sort
– **Success End Condition**: A scrap is saved into a palette for re-use
– **Failed End Condition**: The scrap is not saved, or the scrap is not retrievable.
– **Primary Actor**: Designer
– **Trigger**: The user makes a sketch into a scrap for saving into a palette.
- **MAIN SUCCESS SCENARIO**

1. The user makes the desired sketch into a scrap
2. The user drags the scrap into the palette
3. The palette displays the scrap in one of its boxes

- **Extension**
  4. The user drags a palette onto the canvas
  5. The scrap is drawn on the canvas
  6. The scrap is dragged to the desired location
- **RELATED INFORMATION**
- **Priority**: Secondary
- **Performance Target:** The scrap should be saved and copied instantly
- **Frequency**: Varied
- **Channel to primary actor**: Either of computer, tablet, or whiteboard
- **SCHEDULE**
- **Due Date**: Release date.

Quality Requirements – *Forming the "desired quality characteristics of the system." [6] "A translation of customer needs into a set of quantitatively or qualitatively stated requirements for the characteristics of a product or service to enable its realization and examination. The requirements for quality should be initially expressed in functional terms and documented." [7]*

NFR	Collaborative design - users must be able to collaborate with one another using Calico without having to be in the same location.
Rationale	User-Focused Design. Users have a need to be able to communicate when they are in entirely different locations, be it for health related issues, personal problems, time constraints, etc
Satisfaction Criterion	Users to be supported is maximum 20.
Measurement	20 users shall be connected to Calico simultaneously, and Calico should run smoothly under these conditions.
Risk	Lose team-based clients, and cost opportunity loss from losing innovative feature

NFR	Minimal learning curve - users should be able to pick-up how to use Calico with minimal exploration/learning time.

Rationale	User-Focused Design. Users don't want to waste time learning a new software when they can simply use a whiteboard.
Satisfaction Criterion	Users should be able to learn the use of Calico well within 20 minutes of active use.
Measurement	Users should understand the use of all the feature sets within the time frame.
Risk	Lose users looking for easy to use application

NFR	Fast load times, Calico should start up and work quickly because design thoughts can be spontaneous, and the tool needs to be ready for those moments. Plus users don't like waiting.
Rationale	Saves a bit of energy, and the user is not frustrated from having waiting.
Satisfaction Criterion	All processes should have a time limit. Load time should be under 3 minutes. Casual usage features should be seamless (drawing, focusing, tool selection, highlighting should be seamless.
Measurement	Average time (n=100) shall be taken for all tasks.
Risk	Lose users that become frustrated with load times.

NFR	Secure communications. Any communications between systems should be encrypted.
Rationale	Users need their private documents to remain private
Satisfaction Criterion	All outgoing and incoming transmissions should be encrypted
Measurement	256-bit encryption on all communications
Risk	Confidential data of clients may be compromised. Users will not want to transmit

	under an insecure connection.

NFR	Energy efficient - the software should minimize energy waste.
Rationale	Green sustainability stakeholder requirement
Satisfaction Criterion	Energy consumption should be compared to other software design tools
Measurement	The total isolated energy use of Calico should be 50% less than competitors. Should include driving and flying consumption as well
Risk	Increased environmental pollution

NFR	Seamless - Calico features should build on each other and not interfere with one another
Rationale	User-Focused Design - users don't want the software to have troubles of its own
Satisfaction Criterion	Time spent trying to make a feature set work should be minimized
Measurement	Average time should be kept under one minute.
Risk	Users can be dissatisfied or frustrated about how the features work together

NFR	Minimal bugs - the software should do what it's supposed to do
Rationale	Users don't want the software to unintentionally ruin their sketches or slow down their design flow to deal with unintended behaviour.
Satisfaction Criterion	Total number of bugs reported shall be kept to a minimum
Measurement	# of new bugs/month submitted

Risk	Lose clients on frustration

NFR	Visual appeal - Calico should look nice
Rationale	User-focused design - users are attracted to beauty
Satisfaction Criterion	The number of users that visually prefer Calico over competitors
Measurement	Averaging 4/5 people should prefer Calico
Risk	Cost opportunity loss from potential clients that bse purchases significantly off of visual appeal

Process Requirements – *are the requirements for the development process. The required characteristics of the process project*." [6]

- Because features of Calico must build on each other and not interfere with one another, features are added incrementally
    - Various testing and evaluation done through out
    - Feedback from research is built upon
        - Calico advances in version as features are added, and issues resolved
- To be written in Java 7.0 using Eclipse
    - Reason: Unknown. Choice of developer.
- Testing on Hitachi FX-Duo Starboard
    - Reason: unknown. Choice of developer.
- Asus EEE121 tablets used for development as well
    - Reason: unknown. Choice of developer.
- Analyze and compare other sketching tools relevant to Calico
    - Reason: to get an idea of what features are strong in other software, and what their weak points are to build on them.
- Identify potential future work that could build on Calico
    - Reason: for future development

Deployment requirements – *How the software should be deployed. It's the activities that make up the use software. Activities occur in the producer and consumer side. It's the general process that is unique to the characteristics of each requirements.*

- Must be user licensable
    - Reason: as an access point to users and how the company measures prices and services.
- Must be downloadable (not CD)
    - Reason: to save on production costs, and to avoid using consumables

- Works on all pen-based touch-based devices with Windows, Linux, or Mac OS
    - Usable through a projector, electronic board
    - Reason: to be accessible in a wide range of platforms for the ease of use of variable customers

System constraints – *Contain the restrictions or constraints on how the software should be built. System-related restrictions that don't necessarily results from functional goal. [6]*

- Mouse disabled on devices
    - Reason: to force the use of Calico as a whiteboard, not a computer
- Pen-type input must be used
    - Reason: same as above. A pen is a better tool for drawing than a mouse.
- Use client-server architecture
    - Reason: to enable communication between platforms
- Use own binary format(Calico Packet) instead of other more verbose messaging protocols (XML, JSON)
    - Reason: more compact than original
- Client interface designed using Piccolo2D architecture
    - Reason:
        - lightweight footprint
        - source code available
- Software must be modular
    - Reason: to allow for easy addition later on
- Auxiliary libraries, environments, etc must be open-source/free
    - as per philosophical support, and because of lower costs
- System must use minimal resources in order to waste less energy
    - Downloadable versus CD usable
    - Efficiency of implementation
        - Calculations running on platforms rather than on server

Conclusion

I think writing the requirements specifications for Calico really helped me grasp the idea of what a requirements specification should be. I used to think such things were boring, and that they weren't really necessary, but now I see their true value, especially in their potential to shape the sustainability of the whole technological ecosystem. Ordinarily though, i think the value that comes from requirements specifications is from its ability to layout the whole plan for a business, and even help it acquire funds, and plan out a budget. It really takes care of all the nitty gritty details, and that is an asset to a business which can because of it start off running on take off.

In writing these requirements I will take note that I had trouble say that along the way i had trouble adding sustainability to the requirements. I tried to at times but it felt like I was just sprinkling environmental jargon here and there. And it was also quite abstract. I really couldn't get my head around how i could be more explicit with it. That was until I decided to just make a sustainability plan. This really freed up my creativity and gave me enough room to write everything with my only concern being sustainability, instead of other requirements for example. It was one of the most fun things i had to do for the requirements specification because it made me think deeply and in a broad context. I still think more can be added because I am no sustainability expert.

But i do commend that a guide be made that facilitates creation of these plans through elaboration of the possible sustainability factors for any software system in general and the

enforcement or measurement of these factors. I think this can be a great addition and propeller for sustainability because I don't think Requirement Engineers would hire sustainability experts. So giving them an easy way to add sustainability to their requirements would really facilitate the adoption of the non-functional requirement.

The reason I put the sustainability plan at the top near the business goal (as opposed to with the other non-functional requirements) is because I think at some point sustainability will be a very important aspect to requirements specification. A sustainable economic plan is as important as the business goal itself because a business that doesn't last is a business that failed and lacked planning. I think it was a very smart idea to encompass environmental sustainability with other goals such as economic sustainability, because with it engineers would be more likely to consider the whole package, which is necessary. The importance is there because when the world becomes more environmentally conscious one would want to make the greatest impact. Technology is everywhere, and if we could make software green across the board, it would really go a long way.

Other things I had trouble understanding was the different topic meanings for a system requirement, like the Objectives and Domain Model. When I first started I would search the Internet for the meanings and I couldn't really find anything worthwhile. Luckily, I was guided to the slides. But even then they seemed rather abstract all the time, except the most obvious ones, no matter how many times I read them. What helped me the most was looking at the templates for making them. I suppose you can't teach system requirements without examples, just like mathematics.

As for improvement on my work, I'm a firm believer that for anything, everything can be improved. But to be more specific, I think the constraints can be improved. I really don't know how to add constraints to a software system. In other classes where we talked constraints we merely touched upon them. So i guess i still have a rudimentary idea of them. I based my requirements strictly on what I read from the Calico papers. Other than that, the business analysis i think was really hypothetical and i'm 100% sure my calculations are not correct. I think even professionals get the calculations wrong, especially with software, so I'm not too worried. It was fun making those. Over all, I think i did a good job with my first requirements specification. It was a lot more work than i thought it would be when I first thought about it. Working with google docs and draw.io was a bit of a head ache in terms of formatting, and such. I think it slowed me down quite a bit more than I would have liked. I don't think I'm using those tools ever again. But on a good note, I am proud of my accomplishments.

Works Cited

[1] Cherubini, M., Venolia, G., DeLine, R., & Ko, A. J. (2007, April). Let's go to the whiteboard: how and why software developers use drawings. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 557-566). ACM.

[2] Mangano, Nicolas Francisco, and Andre Van Adviser-Hoek. "Calico: an early-phase software design tool." (2013).

[3] Raturi, A., Penzenstadler, B., Tomlinson, B., & Richardson, D. (2014, June). Developing a sustainability non-functional requirements framework. In *Proceedings of the 3rd International Workshop on Green and Sustainable Software* (pp. 1-8). ACM.

[4] Penzenstadler, B., Raturi, A., Richardson, D., & Tomlinson, B. (2014). Safety, security, now sustainability: the non-functional requirement for the 21st century.

[5] Penzenstadler, B.(2014). Infusing Green: Requirements Engineering for Green *in* and *through* software systems.

[6] Penzenstadler, B. (2014) Informatics 113 Requirements Analysis and Engineering Lecture Slides. https://eee.uci.edu/14w/37030

[7] bizmanuals ISO Definitions. http://www.bizmanualz.com/iso-9001-procedures-templates/iso-definitions

Glossary

Orders of Effect:

First Order: Direct effects of use of a software system. They are immediate and localized. [3]

Second Order: Indirect effects of use of the software system in it's application environment. [3]

Third Order: Broad scale socio-economic structural changes that occur as a result of use of a software system. [3]

Sustainability Dimensions:

Economic Sustainability: Maintenance of economic capital. [3]

Environmental Sustainability: Maintenance of environmental capital, including preservation of resource and protection of the environment.[3]

Human Sustainability: Maintenance of human capital. Health, education etc. [3]

Social Sustainability: Maintenance of social capital. Focus on community continuance. [3]

Technical sustainability: long-time usage of systems and their adequate evolution with changing surrounding conditions and respective requirements. [3]