Design Handbook

Zhuowei Zhang

April 2014

Contents

Part I

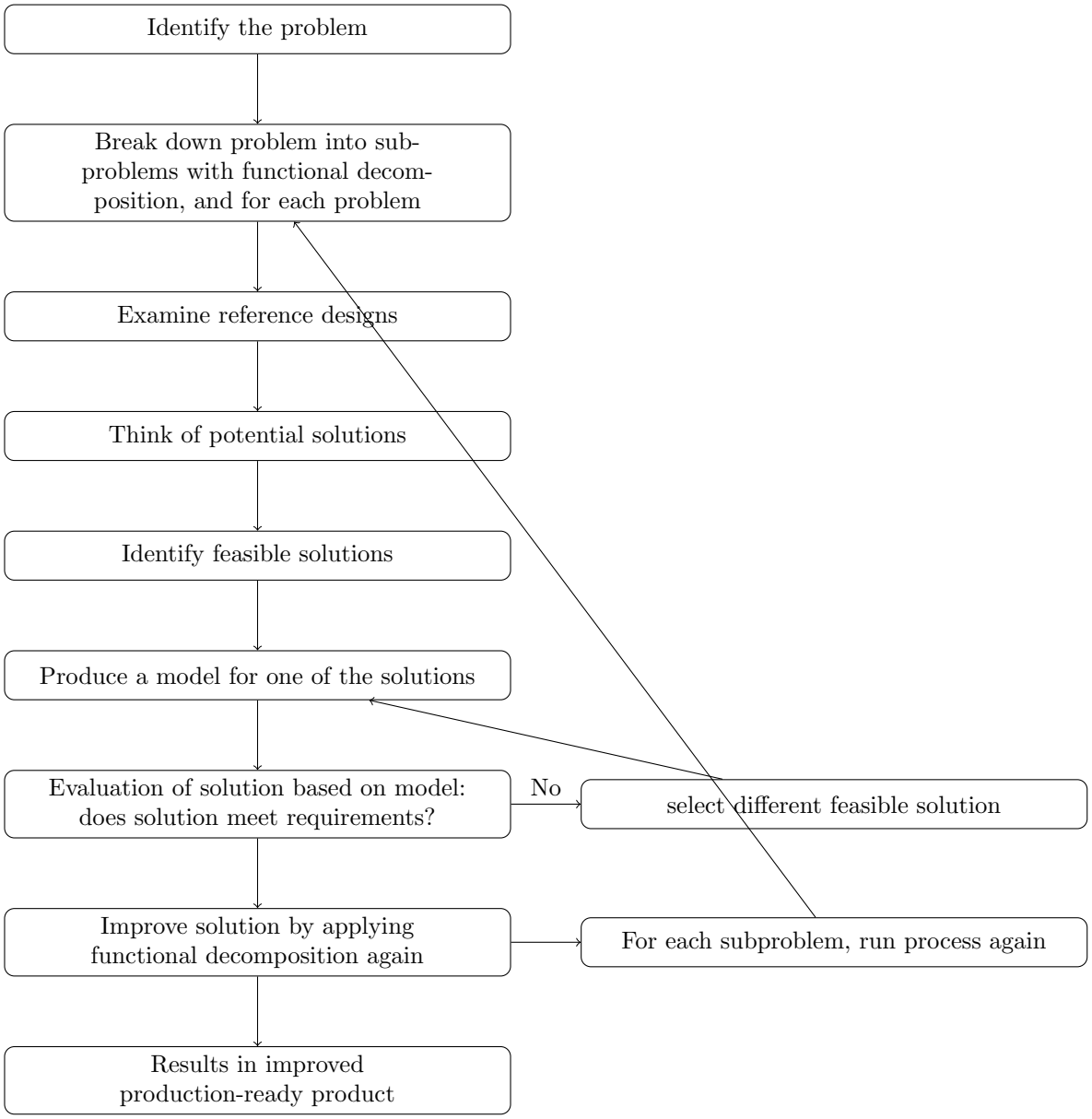
Engineering tools

1.1 Engineering process

My engineering process is a bit unique, as most of my early design work was related to software. A detailed description of my engineering philosophy can be read on the "What I do" section of my portfolio. However, that page is a snapshot of my process at a given time, and does not reflect the evolution of my workflow.

After taking a year of design courses, my design process did not fundamentally change, but was incrementally improved. After learning about the different design strategies, I am able to recognize some of the processes I use in design, and have become more conscious of my actions. For example, I now notice when I'm defining metrics for comparison or breaking down a problem into manageable chunks with functional decomposition. In addition, from experience, I've learned to place extra emphasis on different stages of my design workflow. Prototyping, one of the tools I rely on in my design process, takes longer for the projects we covered in Praxis I and II, and so it now plays a smaller role. In contrast, in Praxis II, out team were tasked to improve a process with multiple steps instead of my typical simple one-step processing tasks, so we spent more time working on functional decomposition to analyze the situation and create solutions targetted to each part of the process.

A flowchart summarizing my design process can be seen on the following page:


Identify the problem
Break down problem into sub-problems with functional decomposition, and for each problem
Examine reference designs
Think of potential solutions
Identify feasible solutions
Produce a model for one of the solutions
Evaluation of solution based on model: does solution meet requirements?
No
select different feasible solution
Improve solution by applying functional decomposition again
For each subproblem, run process again
Results in improved production-ready product

Explanation of purpose and result of each step:

- Identify the problem: my definition of Engineering Design dictates that a work of engineering design must solve a problem. The problem could be something encountered in my own daily life (e.g. when I write on my tablet PC, my palm triggers the touchscreen), or something from an external community (e.g. Praxis II communities' needs/wants, BlockLauncher users' need for a way to make custom monsters in Minecraft)

- Break down problem into subproblems: Most engineering problems you get from step 1 would be vague and overly broad: breaking down a problem allows me to work on one part of the problem at a time. I added this explicit step during Praxis II, when it was a defining part of our group design process: our group took the long process of Scott Mission donation handing, broke it down into stages, and created solutions to speed up each stage. Even before then, I would informally split up problems: for example, "building custom monsters" would be decomposed as "method to render the monster", "method to save custom monsters", and "method to control custom monsters", and I can work on one goal for each release of my app.

- Examine reference designs: when I program software, often I would adapt existing code for my task — such as using the existing twitter-ebooks package to power my Twitter bots. Similarily, we were able to take advantage of reference designs in Praxis II and repurpose them: two of our designs use off-the-shelf anti-slip tape and a lightly modified conveyor belt, respectively. Usage of existing designs reduces time to market - which is one of my goals.

- Think of potential solutions: find various methods that have a potential of solving the problem: this is the divergence, or ideation, stage.

- Produce a model for one of the solutions: often it is difficult to evaluate a solution from very low fidelity prototypes (see the prototypes section for pitfalls I encountered), so medium-to-high fidelity prototypes are useful to test feasibility.

- Evaluation of solution based on model: Now that a prototype exists, convergence can be done by measuring the prototype with the metrics to judge suitability. If it fails, pick another solution and try again. For example, I found out after implementing live patching in my BlockLauncher app that it fails the goal of being able to patch all existing patches, so I chose to implement prepatching instead.

- Improve solution by applying function decomposition again: this stage is dedicated to refining the prototype to production status. The second function decomposition stage is needed because one would only realize the existance of some details after evaluating the prototype. For example, after

implementing the patching feature in BlockLauncher, I decomposed it to more parts: actually patching the game, and selecting patches. Then, I was able to focus on the user experience of importing patches, and was able to change the sorting order of the file browser to make recently downloaded patches more accessible - a problem not foreseen in the original problem.)

- Shipping product: now that the prototype is able to solve the problem, allow it to do its job! As a software developer, this step would consist of deploying the polished prototype as the release of the software. In other situations, such as Praxis I and Praxis II, the prototype would be presented to potential investors and shareholders.

1.2 Prototypes

Prototypes serve many different roles within product design. Prototypes, depending on their purpose, would be constructed to different levels of fidelity: how closely it resembles the shipping product.

1.2.1 Low fidelity prototype

Materials do not have to match actual product: for example, waving hands around to demonstrate a motion counts as a low fidelity prototype
Usually done at the beginning of the design process, i.e. ideation stage.
Very easy to construct. Require no materials.
Purpose:

- Communicate concept
- Evaluate feasibility and potential of the concept

Pitfalls:

- at this earlier stage, some problems with ideas cannot be easily seen. Ideas may be easily prototypeable, and look impressive, but not very feasible. For example, in Praxis II CDR, one of our ideas was a truck with a turntable inside; we built a cardboard mockup that appeared quite cool, but after careful thought and discussion we learned that turntables of that size are hard to manufacture and turn.

Personal examples:

- The aforementioned handwaving in our Praxis II design - allowed us to comprehend the design
- For my Twitter bots: testing effects of enlarging eyes by opening pictures manually in a photo editor and zooming up the eye - proved that effect was grotesque and funny
- For the bird animation: setting the wallpaper of my computer to a GIF manually to see whether animation plays i.e. idea feasible - learned that desktop wallpaper doesn't play GIFs: eliminated idea, avoided wasting time

1.2.2 Medium fidelity prototype

Materials come close to actual product: e.g. using rubber-backed carpet to simulate rubber conveyor in our Praxis II
For testing approaches, factors important to the test being conducted would be replicated; unrelated aspects ignored
Frequently, medium fidelity prototypes for showcasing would take the form of a 3d-rendering or concept art.
Purpose:

- Verify concept
- Solicitate feedback and criticism
- Test various approaches e.g. which type of tape to use
- Used as part of functional decomposition: "stubbing" out unrelated parts of the solution to concentrate on one part
- Demonstrate product to potential investors

Pitfalls:

- For testing, make sure that the choice of simplifications does not rely on invalid assumptions. e.g. my medium fidelity prototype for BlockLauncher kept a copy of Minecraft PE inside the program, since I had assumed that it would be easy to change it to load the game externally. That was, unfortunately, not the case, causing me to severely underestimate from the successful prototype the difficulty to produce my chosen solution.

Personal examples:

- Praxis II CDR: the aforementioned ramp (two versions: both had same angle as real conveyor; one had cardboard top while the later one had the aforementioned rubber; used to select the optimal type of tape and distance between tape)
- Praxis II CDR: the telescoping conveyor leg (built out of a cane; used to prove that telescoping leg would not cause buckling)
- Twitter bots: scripts that tested the face detection: loaded picture from local disk instead of network, but uses same face detection code as real bot
- BlockLauncher: for live patching of the program, hardcoded patches inside the program before working on the patch loader to test whether the approach would work reliably
- BlockLauncher: before making the Renderer API, I tested out feasibility by calling the existing C methods in Minecraft to get a feel of what the API should provide

- Before I write documents, I often devise an outline, which is a medium fidelity prototype I can use to gather feedback and get a sense of the research required

1.2.3 High fidelity prototype

A high fidelity prototype attempts to get as close as possible to the actual product that one would want to produce - with similar materials, dimensions, etc.
Purpose

- Demonstrate product to potential users/consumers (who may not like crude concepts but can easily engage with an almost-production ready prototype)
- Demonstrate production feasibility (because materials/layout faithful)
- Evaluate the product as a whole (as opposed to evaluating individual parts in medium fidelity mockups)

Pitfalls:

- High fidelity prototype does not mean end of development. I often end up shipping what amounts to functional prototypes in code. e.g. settings screens in BlockLauncher was quite messy because I never iterated the design past the point when it first worked

Personal examples:

- Most of my software projects are released unchanged from the high fidelity prototype: but see the first pitfall for why this may be unwise.
- BlockLauncher Entity Renderer API: high fidelity in the sense that it works, but still prototype in the sense that it's still open for criticism/refinement

1.3 Ideation tools

Tools used in the divergent stage of design to generate possible solutions. There are many resources online that contains lists of ideation tools (one guide shown in class was the "27 Creativity and Innovation Techniques Explained" by CreativityToday.net) ; this section will only evaluate the tools I used personally in the aforementioned guide.

1.3.1 Challenge assumptions

Purpose:

- identify unique solutions using points of view that are usually not adopted.

Personal examples:

- After researching the RFP, we asked ourselves what would happen if there was no need to handle donations. From there, we were able to pivot slightly to focus on removing the need for the truck to handle donations - which lead to the curved conveyor belt that aims to drastically reduce the time it takes to maneuver the truck

Pitfalls:

- It's easy to get too carried away with challenging assumptions: we joked about placing a sign saying "no donations", which would still technically meet all requirements but would not actually improve the donation workflow, before we were able to look at the process from a more serious view again.

1.3.2 Classical Brainstorming

Purpose:

- Quickly generate a wide variety of solutions that range in area
- Allow easy group input and feedback on ideas

Personal examples:

- During Praxis I, we used a variation of this where we each sketched out solutions on pieces of paper, and then presented them to the group.
- During Praxis II, we generated a few wacky ideas using this method: the ideas can be seen on my design portfolio.

Pitfalls:

- In Praxis II I discovered that suggesting silly/downright stupid ideas, while fun, was not helpful — I was naively hoping that some of the ideas could be inspiring, but they were so wacky that there wasn't any way to adapt them.
- The person who came up with the idea must be able to explain it: during Praxis I, we were confused by some of the proposals because they were presented without adequate aids to understanding: awful diagram and explanation. (*cough*IwasthepresenterthereIfailed*cough*)

1.3.3 Reverse Brainstorming

Identify how the problem is caused, not how it can be solved - and then the solutions come to you.
Purpose:

- Useful for defensive programming: fix crashes by finding weak points for massive damage
- Useful for user interface design: find all areas that can be confusing, identify reason, reverse

Personal examples:

- This strategy was mostly used to fix bugs in BlockLauncher by identifying potential areas that may break and sandbagging those areas.

Pitfalls:

- The person brainstorming may not actually be part of the community, thus, may not identify all areas of problem - e.g. ui design: I can work user interfaces fine, but some users of my app are less proficient, so I cannot improve the program just by trying to find areas that are difficult for me to use

1.4 Decision making tools

Tools used in the convergent stage of design to eliminate weaker solutions. Again, there are many resources online (such as the list from Studio 8 of Praxis II), but I will only evaluate the ones which I had experience with.

1.4.1 Pugh chart

Multiple criteria tool: takes one reference design, and compare other designs with said design to see if new design better meets each of the defined metrics compared to the reference
Advantages:

- Easy to use: just compare against reference design
- More helpful when multiple solutions are far along in development such that metrics can be estimated
- More helpful when solutions are developed in isolation then brought together (e.g. Praxis I: we each focused on one detailed design before choosing final design as a group with Pugh chart)

Disadvantages:

- Difficult to choose effective reference design: if chosen wrong, everything would be better/worse in all metric and data is useless
- Requires weighting of each metric as a contribution to total score: this weighting is very subjective (e.g. is safety more important than cost in this chart?), and may also require yet another decision making tool to weigh metrics, which increases usage time.
- Requires an extended amount of time per usage, as each metric must be evaluated against each design - limits usage to later stages of the iteration process, where there are less design candidates

Past usage:

- Used in the Praxis I CDR to narrow down the detailed concept designs from four to two: those two were then merged to create the final design. Up to that point, each member of the team worked on one concept design, so we were informed on each solution's projected performance on each metric, and so the Pugh chart was a good fit.
- Used in the Praxis II CDR to justify design selection in the first and second design iterations. This time, the Pugh wasn't a good fit: we used the Pugh during an early stage of the iteration process, which means that we did not know how well some solutions would accomplish their goals as measured by the metrics, and additionally, we were all working on the solutions together, which meant that informal discussion of each solution was more efficient than going through a rigorous process.

Example:
This is the Pugh chart we used during Praxis 1 CDR to rank our four solutions:

Reference: Divided pot	Concepts Hybrid	Concepts Rotating Lid	Concepts Flaps Inserts	Concepts Redesigned Pot
1. Ease of Use	0	-	-	0
2. Portability	0	+	+	0
3. Ease of Manufacture	-	-	-	-
4. Compatibility	0	+	+	0
5. Safety	+	+	+	+
6. Cost effectiveness	-	+	+	-
7. Pouring Efficiency	0	-	-	0
8. Lifetime	0	-	-	0
Sum +'s	1	3	3	1
Sum 0's	5	0	0	5
Sum –'s	2	5	5	2
Net score	-1	-2	-2	-1

First, choose reference design: it shouldn't be obviously awful (since you'll just get useless +s in the table) or the best (since all other solutions would be -s on the table). In this case, we chose the divided pot solution as our reference design.
Then, list all metrics on the left side of the table.
Afterwards, for each solution, compare each metric against the reference design; write + if the solution is better at furfilling the metric than the reference design; - for the opposite, and 0 if they are equal. For example, the hybrid pot is less cost effective than the divided pot, so it received a - in that cell.
Finally, total the scores and sum them up.
Note that this score isn't magic nor should you blindly choose the item with the highest score: as stressed in the Praxis I lecture, they are not set in stone and should be interpreted according to each metric's importance.
In our case, we decided to weigh the safety metric as being the most important (since decreasing injuries was the overall goal of the project). From this weighting, we can see that all the alternate solutions are safer as measured by our metric of volume of liquid splashed out. From this, we thought about combining the reference solution with some of the alternate solutions to keep safety high while improving their other characteristics, instead of just picking the two with the highest net score and continuing.
In the actual Pugh chart, another section was dedicated to justifying each of the comparisons. Due to space restrictions, this isn't shown here, but there would be one short paragraph explaining our choice of score for each metric.

1.4.2 Pairwise Comparison Matrix

Single criteria tool: compare each solution with every other solution, and writing a 1 if the solution is superior to the alternative.
Advantages:

- Also easy to use: just repeated comparisons with other designs
- Helpful for ranking: one usage was to rank metrics based on their importance for Pugh charts

Disadvantages:

- comparisons may not commute: i.e. A compare B gets different results sometimes from team members than B compare A. Confusing to analyze.
- Because of its nature as a single criteria tool, it cannot be easily used to compare solutions against the requirements, which would have multiple metrics defined
- many possible variations: some versions do not fill out the top half of the triangle, some versions have +/-/0 while some versions only have -/0

Past usage:

- Used in the Praxis II CDR to rank criteria for weighting of Pugh charts. Was not very helpful, as the comparisons were very arbitrary (what metric am I using to compare the significance of truck utilization versus facility utilization?)
- Tested as an exam question on the Praxis I final exam. A pairwise comparison matrix with non-communicative comparisons were given to me, and I was asked to interpret it.

Example:
This pairwise comparison matrix was used to rank each objective in our Praxis II RFP to devise a weighting scheme for Pugh charts.
Pairwise comparison matrix of objectives:

	Safety	UtilF	Maint	UtilT	Chem	Load	Cost
Safety	X	-	+	-	-	-	-
UtilF	+	X	+	+	+	-	-
Maintain	-	-	X	+	-	+	+
UtilT	+	-	-	X	-	-	+
Chem	+	+	+	+	X	+	+
Load	+	-	+	+	-	X	+
Cost	+	+	-	-	-	-	X

First, list all the items to compare vertically on the left of each row and horizontally on top of each column.
Then, for each row, compare the item on the leftmost of that row with each item on top of each column, noting down the result in a cell.

For example, maintanance was considered to be less important to our core objective than safety, so it get a minus on the row containing "Maintain" and column containing "Safety".
Finally, tally up the +s and use as a ranking, with higher being better (don't forget the human sanity checking). Our tally is shown below.

Safety	Utilization of Truck	Cost	Maintenance	Chemical Properties	Load	Utilization of Facility
5	4	4	3	3	2	2

1.5 Group work strategies

In the three design courses I attended this term, teamwork was emphasized: in Praxis, assignments are done in groups of 4, while in CIV102, bridge designs were done in teams of 3. During the course of the year, I had the pleasure of working with 4 excellent teams, and was able to observe our strategies and reflect on the strategies that worked well and those that failed.
In this section, I often refer to the two stages of a project: the research stage and the production stage. All of the team assignments in first year followed this pattern: in the research stage, books and standards are consulted and calculations are done, and in the production stage, prototypes are built, documents are written, and blueprints are drawn. Some strategies that work well in one stage will fail in the other.

1.5.1 Online vs real world meetings

In our Praxis I group, we chose to conduct most meetings of the group online, using Skype audio calls for discussions and Google Docs to collaborately edit documents.
In the Praxis II group, we instead decided to meet physically (often in the EngSci common room or the Sandford Fleming Library).
Pros of Skype meetings compared to physical group meetings

- Little to no time wasted to travel to meeting place
- Less distractions than some meeting places e.g. common room

Pros of real world meetings compared to Skype

- Availability of resources at the common meeting place (books if in library, teaching team support and prototype material if in common room)
- Allows for construction of physical prototypes

It's important to recognize the limitations of each type of meeting and use them as appropriate. In our Praxis I group, we chose to schedule physical meetings when needed - to construct prototypes and to research in the library for the RFP.

1.5.2 Calendar and goals

In the Praxis II group, we often had goals of what to accomplish during a meeting. We were introduced to this by the teaching team, when they asked us to fill out cards of what we needed to accomplish during the following week.
The goals helped us focus on our task. At the beginning of the semester, when we were conducting research for the RFP, I found individual goals helpful, since with them I knew clearly what I had to do for the group. Later during the RFP writing process, after the research was complete, group goals, such as

"finish the RFP before Saturday", allowed us to schedule meetings, and during the meetings motivated us to work harder in order to accomplish the goals. Individual goals became less useful, as everyone was now working on the same product (the RFP).
We also found that, without a clear target of what we are supposed to accomplish during a meeting, we always become distracted during meetings, and generally end up doing other courses' assignments. To mitigate this issue, we created a calendar of what has to be done at each Praxis meeting from that day to the end of term, and consulted it each day to assign tasks.
In summary:

- Individual goals more useful in research phase when everyone working on different area
- Group goals more useful in product phase when everyone working on same document
- Set goals for each Praxis meeting with a calendar

1.5.3 Laptops

Pros of laptop usage:

- Everyone can work in parallel

Cons:

- Distraction from social networks

We found that, during the research stage, having one laptop per person was useful since everyone has different tasks. During the production stage, however, only one computer can work on the 3D model or the presentation at any time, and so the other computers become distracting.

1.5.4 Full group meetings

In both Praxis I and the Praxis II group, and even in our CIV102 group, we always did full-team meetups, where everyone meets at the same time. I noted that this strategy keeps team members informed of the progress of the group, and allowed easy exchange of information.
Some other Praxis teams chose to do partial meetups: my roommate's group used texting and Facebook chat in lieu of meetings, and often not everyone would respond immediately. However, my roommate reported that his group had no problem with distributing information of recent changes or coming to a consensus in group decisions, despite the asynchronous process.
In the future, it may be worthwhile to explore chat and other asynchronous techniques that were used by my roommate's group to gauge its effeciveness.

1.5.5 Assignment of roles

For the RFP in Praxis II, our group assigned different roles for each member: one person was responsible for all the writing, two people was responsible for research, and one person in charge of decision making.
Pros:

- Knows task, more focused (has similar effect to goals)
- Takes advantage of each others' strengths (e.g. Balthazar was a good writer, so he was in charge of composing the RFP: I was fairly good at using search engines, so I worked as a researcher)

Cons:

- Some members will idle when there isn't enough assigned task

  During the RFP writing process of Praxis II, after we had sufficient research, the researchers weren't doing much work, while the writer still had a heavy workload. In our Praxis I group, the lack of division of roles meant that a person worked on whichever area needed work the most at the time - I would be researching one moment and proofreading the next.

To avoid the consequences, one can re-assign jobs based o demand - i.e. researchers can become proofreaders as a piece of writing comes closer to completion.