AI Team Organization

Information generally recognized as useful

Vasken Dermardiros

May 2, 2022

As our team is growing, we are finding it more and more difficult to communicate and coordinate the various needs of the team. I've decided to break the team into three specializations consisting of (1) research, (2) development and (3) operations. Research is responsible of exploring new territory and attempting to answer very difficult questions. Development is responsible of taking this code and adapting it to our universal infrastructure assuring scalability and stability. Operations is the main user and requestor of these tools to satisfy the client and its building. Teams being more specialized will require better tailored touch points and cross-communication.
This document attempts to clear our path forward!

Contents

1 Preface

This document will require a shrine in your living room. You will pray to this document before going to bed. Sinners will be shot and eaten.

Date	Version	Changes
18/06/2021	1	First finalized version
18/06/2021	2	Reference to release note added
21/06/2021	3	Fahimeh in AI-Dev
01/07/2021	4	Note about Release branch, typos
13/11/2021	5	Updates teams, projects, resources
02/05/2022	6	Updates links, add other teams

2 Mission

> "Using AI to create value for all building users, owners and utilities across the world."

3 The BrainBox Process

In a nutshell, we qualify a building, install an edge device to communicate with the building – for some building we can connect via software alone –, start extracting data, translate the way they've named things in their buildings to the Haystack standard which we call "mapping" and now we can proceed with the normal business.

First step after is to analyse how the building was operating and propose which sort of control algorithms we should deploy. After confirming with the customer, we slowly take over control of their building gradually, deploying more and more algorithms. Some of these algorithms are rule-based and best practices (ASHRAE Guideline 36), and some of them rely on predictions from our ML models. To train these models, we need to have collected enough data and then, once we have confirgured and initiated the model production, it's literally "set it and forget it"!

As we continue controlling the clients' buildings, eventually we analyse a before-and-after snapshot following a Measurements and Verification methodology. The client convinced, immediately starts paying for our services.

4 Overview

Let's start with the big picture of how the AI divisions fits within the larger BrainBox family.

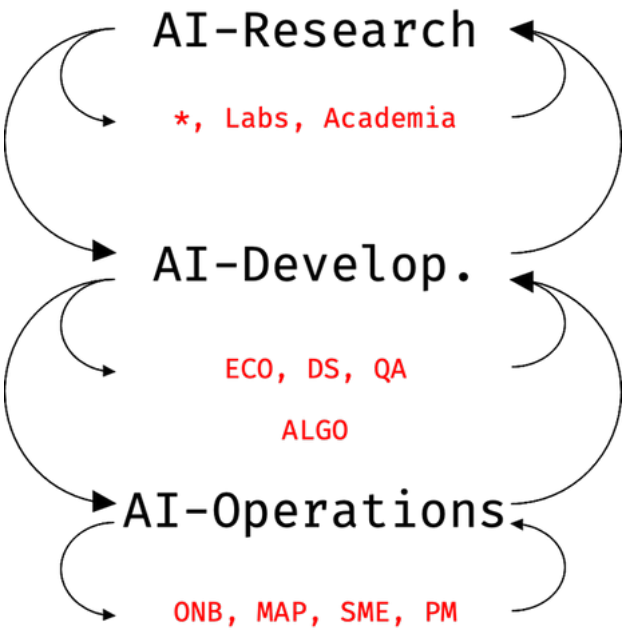

Figure 1: Communication Channels

The **AI-Research** team (exploration) can have discussions with everyone, but would benefit a lot from talking to the SMEs and collaborating with Industry Partners and Academia. They also link up with the AI-Development team to coordinate on how to package the work into something that can be used in production.

The **AI-Development** team serves as the bridge between research and operations. Their role is to assure all code follows the BrainBox standard and fits in the standard methodology set by the QA and Ecosystem teams. They also assure that the code is understandable, troubleshootable and easy to use.

The **AI-Operations** team (exploitation) are the users of the code and communicate with Onboarding, SMEs, Project Managers and others to assure a timely project delivery. The make feature requests to the AI-Development team and, sometimes, to the AI-Research team.

As we go forward, we will rely on **DataStreams** (Lead: Farzam Mohammadi) for data storage and preparation, as well as sending and validating commands to buildings. **Ecosystems** (Lead: Saeid Vosoughi) to be responsible of the framework where all modules will communicate with one another through a microservice-based approach. **QA** (Lead: Elnaz Taqizadeh) to be responsible of developing standardized CI/CD methodologies and hooks, and responsible of enforcing coding standards.

We will focus more on the core work that our team is best fit to do: data analytics, modeling, optimization, anomaly detection, developing techniques to be robust to uncertainties, platforms to version, train, version models and agents, platforms to validate our approaches and simulate the built environment, estimate energy consumption and build virtual meters, estimate greenhouse gas emissions, estimate occupancy presence and preference, and so on.

5 Members

> "Don't let your job get in the way of your work."

This is the proposed team breakdown for the time being. These roles are flexible in the longer horizon. **The members will contribute to their team for at least 70% of their time.**

5.1 AI-Development

- Emilio Botero
- Federico Vallucci
- Fahimeh Hosseini
- Maroun Haddad
- (with additional resources from Ecosystem team as needed)

5.2 AI-Operations

- David Calado-Varela (Team Manager)

- Tien Nguyen
- Minhajul Hoque

5.3 AI-Research

- Emilio Botero
- Scott McDonald
- Fatma Mtibaa
- Claude Demers-Belanger
- Ysael Desage

5.4 Advanced Metering Integration (AMI)

- Marion Blayo (Team Manager)
- Pioter Agalakov
- Jeenu Manivannan
- Jacky Mlakar
- Ray Chen

6 AI-Development

> "If you don't have time to do it right, do you have time to do it twice?"

Role: bridge methods into deployable and robust solutions, ask the right questions, harmony within full dev team

Values: clarity, structure, discipline, foresight

6.1 It's about trust

Figure 2: Will the two parts meet?

Do you trust DataKit? Do you trust ____?

What makes you gain trust? Lose it?

If something is agreed upon and work is done independently until it comes together and it doesn't fit, what happens to trust?

This script always crashes, it's not stable, I don't understand what's going on, it's so obfuscated and cryptic. I can't even troubleshoot it! Should I just code my own version of it? I need to get something working… So annoying! </3

Solution: organize a session where you explain what you plan on doing before doing it. Get feedback at all stages of the work. Don't wait to have something fully complete to push to master. It's too late! For larger projects, write the skeleton first, e.g. a Class that does nothing but lists all the methods therein with their expected inputs, outputs and what is the planned behaviour.

A construction site employs hundreds. Teamwork. There is no "I" in "BrainBox AI"!

6.2 Request Process: AI-Operations <-> AI-Development

With the business advancing, we strive to improve efficiency and efficacy. In plain English, we're getting more and more work to do but less and less time to do it, so we need to work smart or risk burning out. If our work scales linearly, we're screwed.

1. (Oper) A specific need is requested by AI-Operations; if need comes from a recurring issue, Asana ticket references must be provided
2. (Oper) How should it work?
3. (Oper) Is it a function? What are the inputs and outputs? What's the name of the function? Argument list, if possible?
4. (Oper) What's important? Nice to have?
5. (Oper) Might it break another process?
6. (Oper) Submit request as a `DEV_FORM` in Asana
7. (Vasken) Thank you for the request/feedback!
8. (Vasken) Review form and assess priority, time requirement, ask for clarification if needed
9. (Vasken) APPROVE: provide a tentative date or release cycle for completion; REJECT: explain why or provide alternative approaches
10. (Dev) Go over `DEV_FORM`, ask for clarification if needed
11. (Dev) Add project tag to `DEV_FORM` to copy into personal dev plan
12. (Dev) Describe how the deliverable will be approached within the `DEV_FORM`: will you need extra resources? people?
13. (Dev) Stamp a date or release cycle on it
14. (Dev) You better work dev
15. (Dev) Project completed containing an example either in docstring or in `__main__`, and a unit test covering edge cases
16. (Dev) Push to `master`: QA and code review
17. (Oper) Tech-transfer: Receive the code, function or module from Dev, go over documentation
18. (Vasken) Release announcement of new code, function or module
19. DONE.

Example:

1. (David) "I need a way to know if there's a data gap or if it's an extraction issue, I've been having this issue for buildings A, B, C and here are their corresponding Asana tickets

2. (David) Function will need to check in the “extraction_cycle” table for the building and check if the date of the last extraction is too far in the past relative to now
3. (David) Should be a function running within ControlKit Data Module; inputs is the building id and output is an indicator signaling if it’s a gap in the data or if the data hasn’t come in yet
4. (David) Would be nice to customize which extraction list to look for, but I really need it to print a lot of information because I’m sick of checking the tables every time
5. (David) Don’t think it’ll break anything
6. (David) Submit or input `DEV_FORM`
7. (Vasken) Well received! :thumbs_up:
8. (Vasken) Looks doable! Question: by “building id” do you mean the id column in the architecture_dev.buildings table? Yes? Ok!
9. (Vasken) APPROVE! Looks like something not very difficult and would be something needed sooner than later. “Fede, do you think this is something you can take care of? Do you have the time?”
10. (Federico) “Maann! Of course! Everything looks clear!”
11. (Federico) Copy to Federico’s personal dev plan by linking the project. Advance tags in Asana as necessary for SOC2 compliance.
12. (Federico) Function will query extraction_cycle table. Need an SQL credential. DataKit has a method to get the full line for the building in the architecture_dev.buildings table based on the id. The database and table names are there, so I can then find the right extraction_cycle table. From there, I will query the last one, … (You can also describe this as pseudo code. Description can be copied to the docstring afterwards.)
13. (Federico) Should be done by next release cycle
14. (Federico) Work, work, right away!
15. (Federico) I feel like Pablo when I’m working on my code! Nice, clean, code. Nice documentation. Variables that mean something. Unit test passes.
16. (Federico) Time to push to `master`, organize meeting with Vasken, Elnaz and another person from the AI team. Go over description in Asana. See if the code does as is described. If everything is Gucci, finish the pull request (PR).
17. (David) “Wow Fede! Your function is so awesome! You just made such a positive change to my life!”
18. (Vasken) Cool stuff! “In this release cycle, we’re releasing functions x, y, z and modules a, b, c. In a nutshell, they do this and that. Thank you to … for your contributions! Find the documentation here! In the next cycle, we will release this. The one after that covers that. Stay tuned!”
19. DONE.

6.3 How can AI-Development gain trust from AI-Operations

- Request has value
- Tool is used correctly until a better one comes along
- AI-Operations understands its use and limitations

6.4 How can AI-Operations gain trust from AI-Development

- Doesn't break anything
- Does what was requested
- Developed code is easy to use and understand
- Code has been tested extensively for bugs and edge cases (proof? unit test?); excellent coverage
- Delivered within promised time frame or explain why it wasn't
- Better to take longer and deliver quality that doesn't need fiddling around. Rework is very expensive!

6.5 DEV_FORM

NOTE: This approach is deprecated and replaced with the Shape Up methodology for longer-term projects.

I called it a `DEV_FORM`, but I'm shamelessly stealing David's work! We should really use his project as a template **if** we need to make another dev tracking Asana project. Please try to rely on his so it's easier/simpler to track things. The ControlKit Development Pipeline is where the developments are and there are two templates that should be used: (1) the first is more for bug fixes and hotfixes, and (2) the second is for new features and enhancements.

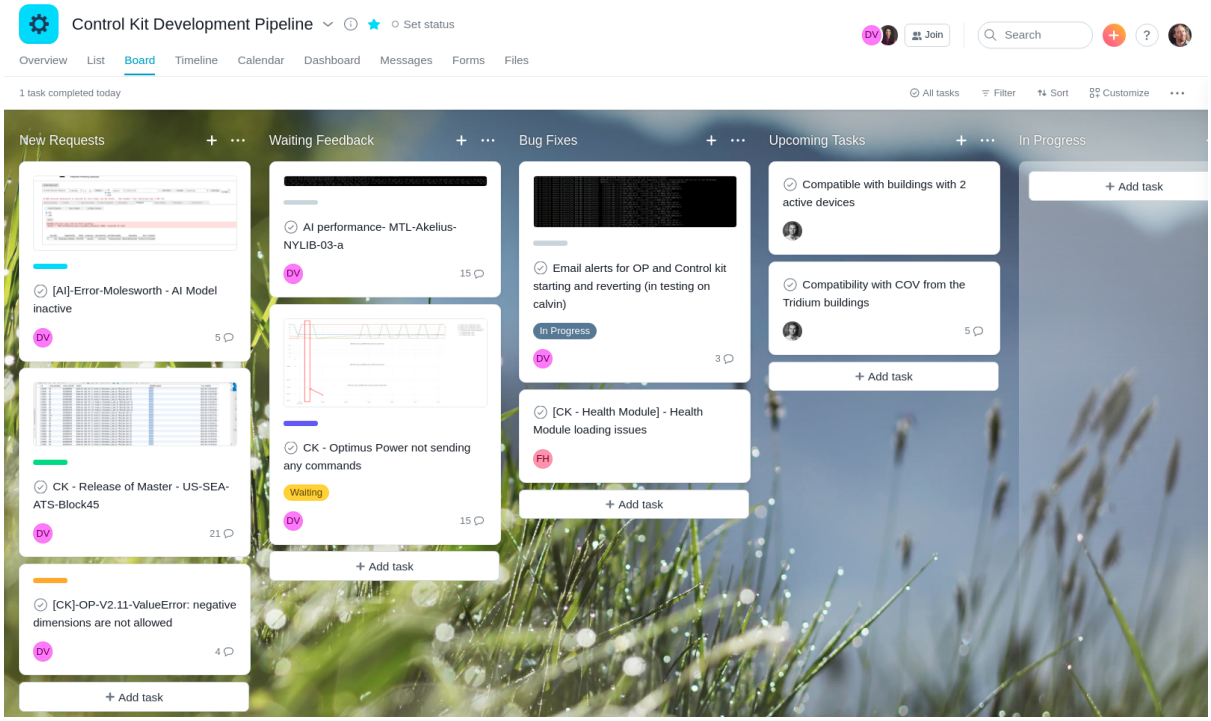

Figure 3: ControlKit Development Pipeline

Not currently active, but we can also add in the `Dependencies`, `Tags` and custom `Asana` (difficulty level described by yoga positions) fields in there. We also worked on an Asana convenience library to extract the info into a CLI and to add tasks via Python if that's your thing.

6.6 Development Release Cycles

Partly to slow things down so that we can appreciate people's work (translation: to do a proper code review), we will release new code and functions every two weeks. These releases will be announced in our [AI-friday] meetings, placed in the Teams release page, documented and transferred to AI-Operations.

Also see the document Elnaz prepared called 'Release_Note.docx'. Release notes will go into a Teams page (to be created by QA).

In the case of emergency **hotfixes**, code can be updated as soon as possible. Note that hotfixes are exceptional cases and should *never* be the norm. For every bug in production, ten kittens are killed. Very cute kittens. Fluffy. Dead. Because of you.

6.7 Git Structure

The accepted company approach is using the master-dev-feature approach, a.k.a gitflow.

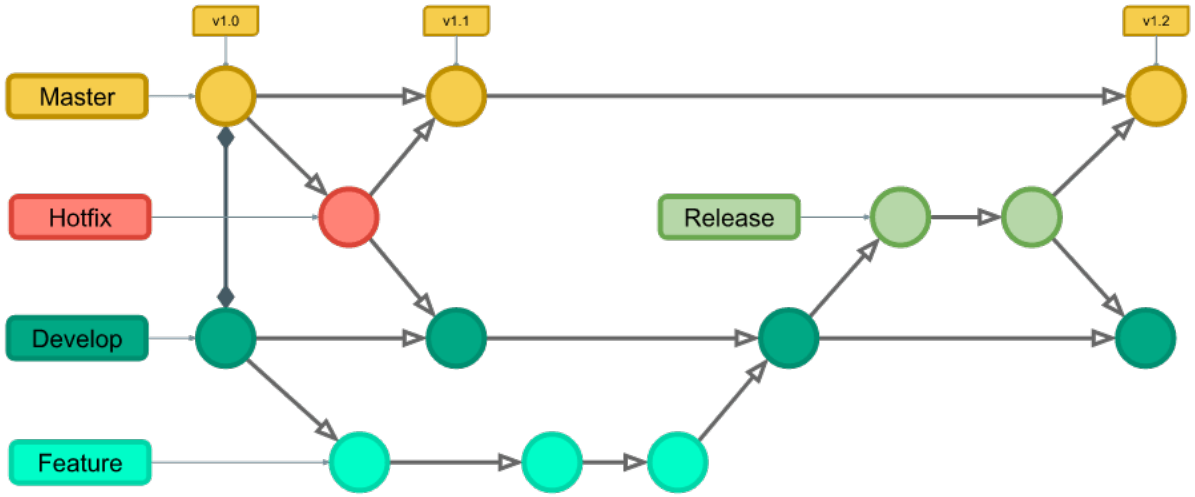

Figure 4: Gitflow diagram

Commiting to `master` and sometimes `dev` requires you to submit a pull request (PR). The contents of the PR needs to follow our standard way.

At every commit to master, a release is issued and the version is incremented. Increments follow the convention: "version `x.y.z`" where `x` represents a major version which can break backwards compatibility, `y` represents a minor update, and `z` is for patches. All merges to master must be rebased to the current version of master to assure hotfixes are not undone.

The git image does show a `Release` branch. Using this is prefered to bundle some features together into one update. Prefer this over having a branch called `dev_master` or `dev_to_merge` or `almost_master`.

Breaking changes are not tolerated. All breaking changes must be announced and will be thoroughly reviewed.

6.7.1 Branch Convention

Main branches:

- `master`: production branch and version incrementally tagged. All merges to `master` requires approuval from QA
- `dev`: parallel to master where features merge into here

Feature branches must have a meaningful name while using one of the following prefixes:

- `feature/...`: adding a new feature or functionality; merges to `dev`, may or may not warrant a code review
- `test/...`: temporary branch to test or troubleshoot
- `fix/...`: fixing an issue or bug; merges to `dev`
- `hotfix/...`: emergency branch to fix an issue in production; only feature branch that can be merged to `master` directly
- `release/...`: (optional) branch from `dev` getting ready to merge to `master`

Branch names **must** be in small-caps. Merge without fast-forwarding by using the `--no-ff` flag.

6.7.2 Commits

- Use the present tense in commit messages
- Write meaningful messages. Avoid: “fix”, “try”, “try again”, “one more time”, “sfsdqwerhqdfsa”
- Use small commits and do it often; you can cherry-pick better
- Never use `git add *` or `git add --all`! Use `git add -u` if you must.
- Create a `.gitignore` file for Python programming, make sure that pesky `.DS_Store` doesn’t creep in if you’re on MacOS!!

6.8 Code Review

Check out QA’s (Elnaz’) notes on the subject. Check out my notes too on the [code_review] process to understand it’s *raison-d’être*.

Generally speaking, write code that is understandable and can be troubleshooted without your help. If you’re aiming for the beach, how can you move there if AI-Operations can’t do anything with your code without you?

Code review is a big commitment. It takes time and planning. Meet and talk with your reviewers if this is your first submittal or if there are large changes. Do chat with them before even starting to code! (See note in previous “Trust” section.)

7 AI-Operations

> "In AI we trust."

Role: apply models, methods, monitor, understand and demand better tools

Values: communication, speed, composure

7.1 Liberation of Stress

- Stress f***en kills. Do you need me to show references?
- Distractions: turn off Teams or mute to get shit done
- Block off time: reject/filter meetings -> ask yourself, how will this meeting benefit me and make my work go better/smoother? it's not? dump it and tell them I allowed you
- Block off your lunch: go out for a walk, play frisbee with your dog
- **URGENT** stuff -> fix it, but then get to the bottom of it:
  - Does this happen often?
  - Are you okay to work like this?
  - What have you done about it?
  - What can AI-Development do to help out? Are you missing a tool?
  - What can I do about it? Do you need help? Do we hire someone? Hiring process typically takes months, so plan ahead!
  - Do you feel stuck?
- Urgent work $\neq$ Important work: firefighting is urgent, preventing fires is important.
- Count your hours if you have to, build a case.
- **I don't want anyone doing more than 40 hours on average!**
  - Some project will require immediate attention, handle those, but do take off to compensate: verbal agreement.
- **Don't sacrifice yourself.** We're not in the martyr making business – *Mardiros* means martyr in Armenian though, very sus. Know your boundaries and assure others understand them. If the work has to be done by another department, get a commitment (describe the role, get feedback, make them repeat it and highlight the tasks) and respect it. Document it. Stand your ground, but always be respectful.

7.2 How to troubleshoot code

Before copy/pasting a screenshot to AI-Development, make sure it's not a "Code-18". Follow these steps:

1. Read the traceback message
2. Did you do step 1? Go do step 1.
3. Great. Understand the message. What can cause that failure? What are the steps done before getting to that point? Could the error have been because the code was able to pass a loosely written `try`/`except`?

4. Run the code in debug mode to increase the verbosity of the prints.
    - You could use the iPython debugger: `import ipdb; ipdb.set_trace()`
    - Or in VSCode, PyCharm, Spyder: put a red dot at the line to interrupt the code when re-running in debug mode (F5 in VSCode)
5. When code gets to that spot, check the variable values -> are they ok? Reasonable?
6. Read the lines above it. Do you understand what they do? Is it clear? Cryptic?
7. Try a few things:
    - Override the value
    - Try with another input, building or other
    - Modify the code a bit
    - Does it fix the issue?
8. From here on, you can, if possible:
    - BAD: make a public announcement on how bad of a coder the person is
    - GOOD: communicate the issue to AI-Development
    - BETTER: and specify the conditions to reproduce the bug
    - BEST: and fix the bug and prove that it doesn't break anything else by running the unit test! Make a PR. Your next round's on me!

8 AI-Research

> *"Quae Quaestio"*

Role: challenge current state-of-the-art, conceive and build better models, methods and approaches, be up to date with literature, hack techniques together

Values: curiosity, independence, thorough persistence, ambition

8.1 Applied Research: AI-Development <-> AI-Research

What does it mean to be bleeding-edge?

1. No deadlines but requirement to show significant progress or analysis.
2. Projects pursuing very specific performance outcomes, eg.:
    - Level of accuracy
    - Multi-use simulation platform
    - Ease of understanding or interpretability of the model
    - Compositional models
    - Model works with missing values
    - Model works for non-uniform time
    - Model works anywhere!
    - Controller self-adapts, learns online, provable

    - Control approach works anywhere!
    - Self-supervised anomaly detection
    - Data explorer platform: what does the data look like, density, what does the model predict, when is it off, model accuracy QA

3. General and generic approaches, or works on certain class of problem, application scope/limitations
4. **Failure is great!** Failure has a very clean signal compared to success. If failure is documented, we will never make the same mistake twice and will learn to overcome it.
5. Journals, reports, notes, documentation, code snippets, jupyter notebooks and so on must be committed to git

    - Asana forms not mandatory for Research projects
    - Will communicate and share through a Team channel; our **Curator** will be involved!

Need can come from *anywhere* but is much more *thoroughly reviewed* and voted on as a team. Aim before you shoot!

Discussions are obligatory -> **no hermits!** Researchers have to be open to talk, discuss, defend, analyze, read and review papers, design experiments with clear hypotheses and conduct these experiments.

Research is extremely difficult and overwhelming. You should follow that:

- simple > complex (can you defend adding complexity?)
- complex > complicated
- general > specific
- specific > unique
- beautiful > ugly
- elegant > beautiful

8.2 So you have something good...

> "In theory, there's no difference between theory and practice, but in practice, there is."

The research approach looks promising! Let's put it in practice! Time to get the AI-Development involved!

1. AI-Research explains *key results* backed by a more technical report

    - This report can become a conference or journal publication, or a white paper
    - This part if akin to how AI-Operations makes a new feature request

2. **AI-Development is responsible to package the code** into a *standardized* framework as approved by the whole dev team (lead by Ecosystem and QA)

    - Framework requirements can be communicated ahead of time
    - Doesn't mean AI-Research can get away with shit code!

3. Once packaged, AI-Research to review *performance* of AI-Development's version, implementation or rework

    - Was there a tradeoff done?

4. Dev/QA to review *quality* of code, understandable and documentation; similar to how code requested by AI-Operations is delivered

8.3 So you don't have anything good...

Part of being on the cutting-edge is that we are taking risks. Risk implies failure and failure is good.

How to go about failure? Can you give up? Absolutely! Park it. Document what you have done, how, explain where you stopped and why. You might come back one day or someone else might pick up from where you left off.

8.4 Current Projects

We will be asking for you to define the end of the project. You can't work on something forever.

- Automated anomaly detection
    - Goal: self-supervised anomaly detection
    - Performance objective: [TODO: to fill]
- Gargamel
    - Goal: generic control approach using genetic algorithms
    - Performance objective: quick and easy algorithm to deploy for the general case; archetypes may benefit from specialization to reduce search space
- SwarmAI
    - Goal: hierarchical optimal control based on RL and OR
    - Performance objective: [TODO: to fill or link to project]
- Simulator of the built environment
    - Goal: have the next best thing to a real building or building cluster
    - Performance objective: have various models for different objectives. [TODO: add details or link to project]
- Self-install for RTU buildings
    - Goal: plug-and-play solution for RTUs
    - Performance objective: from auto-mapping to auto-controls to auto-monitoring to be done with very minimal human labour and should be software based; link with [autobot]
- Autobot
    - Goal: automatic mapping of buildings using haystack tags
    - Performance objective: automate by large the point selection and naming processes; rely on templates, data and data co-variances

- Composable physics-based or physics-inspired models
  - Goal: towards interpretable grey-ish-box models that will perform well in extrapolation and with missing data
  - Performance objective: composable approach where submodules can be more grey-box given information or data or swap out as black-box is less information is known or if performance justifies; think of it as designing a "building submodel modelling grammar" where a building is composed of 10 of those models and 3 of those
- MLOps model factory and model serving
  - Goal: train the models in an isolated environment in ClearML and to allow for experimental work. Longer-term goal is to serve models directly from ClearML
  - Performance objective: push as many new model requests as possible and train them all without fail. If failure does occur, inform involved parties and provide enough information to make troubleshooting quick and easy.
- Mission Control
  - Goal: the loving face of Control Kit
  - Performance objective: to allow to monitor and interact with Control Kit without having to go through a CLI. Able to check past predictions, data and other information in a convenient way and to allow downloading this data locally.
- ControlKit-as-a-Service
  - Goal: choke the ControlKit from a monolith into services
  - Performance objective: make all the modules within ControlKit become decoupled and independent. Communication in between with APIs.

9 Touch Points

> "Meetings are deadly."

But we're online and we need to somehow tell people to meet us somewhere in the path of work. On the AI team, we only have our Friday meetings that are fixed. Those are 9-10 AM Eastern Time. In there, we talk about big progress, tech issues, code best practices, invite members of other teams, get updates from the research team and so on. It's not very formal.

Other touch points are Lunch and Learns and the TownHall which is organized by HR.

10 Secondary Roles

> "If it's everyone's responsibility, then it's no one's responsibility."

Secondary roles is me asking you to help the team. These things should not consume more than 5-10 hours a month.

- The **Curator**: don't let your memes be dreams. AI Channel moderator. Post articles, news, cool stuff, whatever
- The **Code-Master**: brings in coding best practices, tips and tricks to use git for debugging (+ cherry picking), troubleshooting, package managing, CLI tools, scripts to set up working environment, docker
- The **Librarian**: organize BBAI wikis, references, cheat sheets, code snippets to make it accessible for AI team
- The **Ambassador**: the face of the AI team. Attends the introduction meetings (1 hr meeting, twice a month) or any other meeting requiring AI blessings
- The **Lecturer**: prepares and presents small lectures introducing subjects like ML, RL, optimal controls, HVAC, and so on either to the team or in the Lunch and Learns; between Ysael and I, we have quite a bit of material! Much more Ysael at this point.
- The **Storyteller**: recites anecdotes of past experiences, of funny or not so funny things that have happened, testimonials of failures and successes
- The **Scholar**: looks for interesting courses and training that can be beneficial to the team, also responsible for following up with expense forms; person also responsible to make sure the reading session is organized and we have a presenter + paper
- The **Organizer**: gets feedback from the team and then looks to invite a guest speaker in the [AI-friday] meeting! Don't forget to confirm with the guest the day before :s
- The **Welcomer**: responsible of making sure new employees get all they need to get started: access to LastPass, Timesheets, Asana, git, DBs, VMs, ClearML, also helping them set up PyCharm, Spyder or VSCode
- The **Perfectionist**: looks at how we're doing things and thinks of ways to improve our workflow, communication and use of time. Work smart, not hard.
- The **Experience Creator**: design of interfaces, plotting standards, templates for reports, presentations, and so on. Coordinate with Marketing and Philippe our senior front-end developer
- The **Shoji Morimoto**: person that has the role of not having a role

11 My Role as a Manager

- "Control Tower" – JSV as of June 9, 2021
- Unblocker: support for coding, science and technology questions and suggestions, people skills (lol)
- Orchestrator: seeing the pieces come together
- Cheerleader: what I identify as
- Conflict resolution, escalation: will bully for food
- Approve/Reject `DEV_TICKETS`
- Liaison between teams and departments

11.1 Stuff you don't care about but I still felt like sharing

- What do I enjoy? Research and development

- What do I hate? Urgency, shit and bullshit
- Do I need constant motivation? No
- Do I do things I don't enjoy? Limited patience
- If my enjoyment is interrupted, what do I do? take action lol ofc
- I hate being a manager and couldn't care less about titles (PhD included), doing it more out of necessity
- I hate office politics
- I love being surrounded by bright, motivated and positive people
- I love having deep conversations
- I'm a tinkered, hackers, dreamer, curious about everything and anything
- I love seeing cool stuff work, even if it's stupid, actually, it's even better if it's stupid: I made a peanut butter cup dispenser that pushes out a cup filled with peanut butter once a day because my gf is addicted to peanut butter and I got tired of hiding the jar

12 FAQ

> "And don't worry about the vase."

- Q1: Can AI-Research help with AI-Development?
- A1: Yes *iff* the task is well defined and short. Development should not reduce the research work to dip below 70%. End of the day, no one is 100% of any side.
- Q2: Can AI-Research help with AI-Operations?
- A2: Let's say *no* and see what happens
- Q3: Can AI-Development help with AI-Operations?
- A3: Yes! As long as the AI-Development can still do development for 70% of their time.
- Q4: Can AI-Development create something that operations didn't request?
- A4: Of course! Our job is to automate ourselves out of existence! If you foresee a need, go for it!
- Q5: What is considered operations?
- A5: Onboarding a building, chatting with onboarding about a specific building, chatting with clients, conversion meetings
- Q6: Can I put a dev or feature branch in production?
- A6: Absolutely not! To get to production, code must be tested, QA'ed and then can go in prod.
- Q7: How much time should I spend in code review?
- A7: For small incremental changes, not too much time. For larger changes (more than 200 lines of code for example), perhaps it warrants a meeting with the PR requestor to go through it. The code reviewer should understand the code's intent, what it does and how. He/she should be able to run the code and monitor the variables. We are aiming to standardize the interaction of modules and this will greatly facilitate testing; this won't be next week though.
- Q8: Who is aware of the research projects under development?
- A8: The AI team and Jean-Simon. No one else needs to know and can be viewed as a *DEV secret* and/or *AI secret*. DON'T EVER MENTION ANY OF IT TO SALES. THEY WILL SELL IT. DON'T EVER MENTION ANY OF IT TO SALES. THEY WILL SELL IT. DON'T EVER MENTION ANY OF IT TO SALES. THEY WILL SELL YOU. DON'T EVER MENTION ANY OF IT TO SALES. THEY WILL SELL IT.

13 Resources

13.1 New Member Onboarding

Welcome to the AI Team! It's time to kick ass and chew bubblegum!

(Copied over from the [managers_playbook] Onboarding section)

Onboarding material:

1. Team mission
    1. How is your team moving the needle for the customer?
2. Team members
3. Repositories and services
4. Documentation:
    1. Code contributing guidelines
    2. Ticketing process. E.g. label and story pointing guidelines.
    3. Glossary of terms.
    4. Releasing code.
    5. How tos (e.g. migrate database, add secrets)
5. Getting started:
    1. Installation instructions (e.g. Docker, postgres).
    2. Getting the right accesses (e.g. PagerDuty).
    3. Running your apps locally.
6. Meeting setups. Who should your new team member meet with?

- Email
- Login
- LastPass
    - Secure storage of all passwords
    - lastpass.com/
- Gitea
    - Git-based code repository
    - Link to AI Team repo
- TimeSheets
    - Input hours and projects you worked on
    - Link to login page
- BambooHR
    - Human resource platform: info about you and where to request for time off

  - Link to BambooHR

- Asana
  - Project tracking
  - By putting an `at brainboxai.com` email, it'll add you to the company site
  - Link to Asana

- SQL Database
  - Connect using MySQL Workbench
  - Username and password will be added into LastPass

- VPN
  - We use NetExtender
  - IT will help you set up the Authentication tool on your phone

- Cloud
  - Request for SSH access to virtual machines done by your manager
  - You should be getting instructions to set up the Amazon VPN (AWS VPN) from Ecosystem

- ClearML
  - Request for access to ClearML cloud done by your manager
  - Request for access to ClearML on-prem requires you to be on the VPN and have SSH set up with `unicorn` (name of our on-prem machine running the ClearML server)

13.2 Our Machines and VMs

As you can imagine, we have a few machines doing a few things. Some are virtual machines running in Amazon's cloud, others are machines in our office in Montreal.

13.2.1 Virtual machines you might care about

The following are running in the Amazon Cloud and are AWS EC2 instances. To connect to them, you need to be connected via the **AWS VPN**.

- `calvin`: main prod machine running ControlKit 1.0
- `calvin-staging`: test machine running ControlKit 1.0 and where we test changes
- `calvin-{us,au,...}`: clone of `calvin` for other regions besides Canada
- `sim-1`: machine set up to run the EnergyPlus-based virtual building simulations
- `sim-2`: machine set up to run the Modelica-based virtual building simulations
- `mock`: machine running a mocked version of Mission Control (ControlKit's dashboard)
- `unicorn-cloud`: machine running the ClearML MLOps server
  - There are many worker VMs attached to this but you don't need to care about those
  - Link to ClearML-prod: https://app.unicorn.brainboxai.net/dashboard

13.2.2 Machines you might care about

The following are machines in our office in Montreal. They are higher-end desktops with gaming graphic cards. Think Intel i9 or AMD Threadripper with Nvidia GTX 2080s with a lot of RAM. To connect to them, you need to be connected via the **SonicWall VPN**.

- `unicorn`: desktop running a mirrored ClearML MLOps server where we test things before moving it to the cloud. This is the machine you will directly communicate with and it will schedule your jobs out for you.
  - Link to ClearML-dev: http://192.168.20.20:8080/dashboard
- `corona`: desktop running ClearML Agent
- `ditto`: desktop running ClearML Agent and also has the EnergyPlus-based virtual building simulation set up
- `ectoplasm`: desktop running ClearML Agent
- `xavier`: Nvidia Xavier NX SBC running ClearML Agent
- We may add a RaspberryPi in there at some point too

13.3 Video Tutorials

These may require permission to view and because of migration may no longer be working. Let me know and I'll update them!

- AI Lunch and Learn: Overview of what the AI R&D Team is up to! (Jan 26, 2022)
  - Slides
  - Video (second half)
- AI Primer (V.Dermardiros)
  - Slides
  - Video
- Global AI Presentation (Y.Desage), link to video
- HVAC Primer (V.Dermardiros)
  - Video
  - Slides
  - Hand-written notes
  - Simple drawings of air-side systems
- HVAC Primer (D.Calado-Varela), link to video + slides BROKEN LINK
- Previous sales slide deck with Vasken's notes on building and the grid
  - Slides
- BrainBox AI Sales Presentation
  - Slidese

- Mission Control
  - Overview screencast (D.Calado-Varela, V.Dermardiros): link to screencast

13.4 Linux Stuff

- Vim [vim]: Text editor from the 70's that's still the best – come at me bro!
  - You can also check out my vim dotfile, it requires you to install Vundle, the vim package manager. There are a bunch of near packages I use in there! Let me know!
  - Side note: I use spacemacs (and org-mode!) on my personal device. It just doesn't work well in Windows.
- VSCode: Very powerful text editor with a lot of customization
  - Foam: Note-taking extension to better organize your markdown notes and create a second brain
  - You can check our my [vscode-setup] too
  - Talk to Federico, he'll tell you which extensions are useful and interesting
- FiraCode: Typeface I love and enjoy. It also has ligatures!
  - https://github.com/ryanoasis/nerd-fonts
- Git [git]: Version control
  - We use the master-dev-feature approach; see in above document for more info!
  - You can also see Elnaz' git tutorial page
  - Instructions to submit a pull request (PR)
- I prefer using zsh for my linux terminal
  - You can also check out my zsh dotfile, it requires oh-my-zsh to be installed. Again, cool stuff in there!
  - Try the `mark <name>` command to bookmark the location you're in and `jump <name>` to jump to that location! `marks` will print out all your bookmarks.
- Regex [regex]: Regular expression search and filter used a lot of places
- Bash [bash]: Command-line functions
- cron: Job scheduler
- grep, find, awk, sed

13.5 Wikis and Documentation

Please note that we are moving documentation to Confluence.

- Algo: Optimus Prime
- QA Documents
- Release Notes: it's a Word file
- My notes on [code_review]

13.6 Tickets

- Ecosystem DevOps Request (VM/DB access, software, etc)
- Ecosystem Deployment/Software Troubleshooting Request
- DataStreams Software Request
- Onboarding Urgent Tickets (limited access)
- ControlKit Dev Request

14 On the fallacy of the beach

The beach. Ah yes. The beach. The warm breeze hugging your skin, the sun kissing your face while birds fly high and the wave crush down and sizzle onto the sand. Close your eyes.Just for a second. Do you feel it? Look over here, this is the beach. Prestine water, blue as a sapphire, sand as white as ivory.

The beach is where we're going once we've automated everything.

Ok. It's a joke. That'll never happen. But since we're engineers, the following chart is useful to know how much time you can spend automating something!

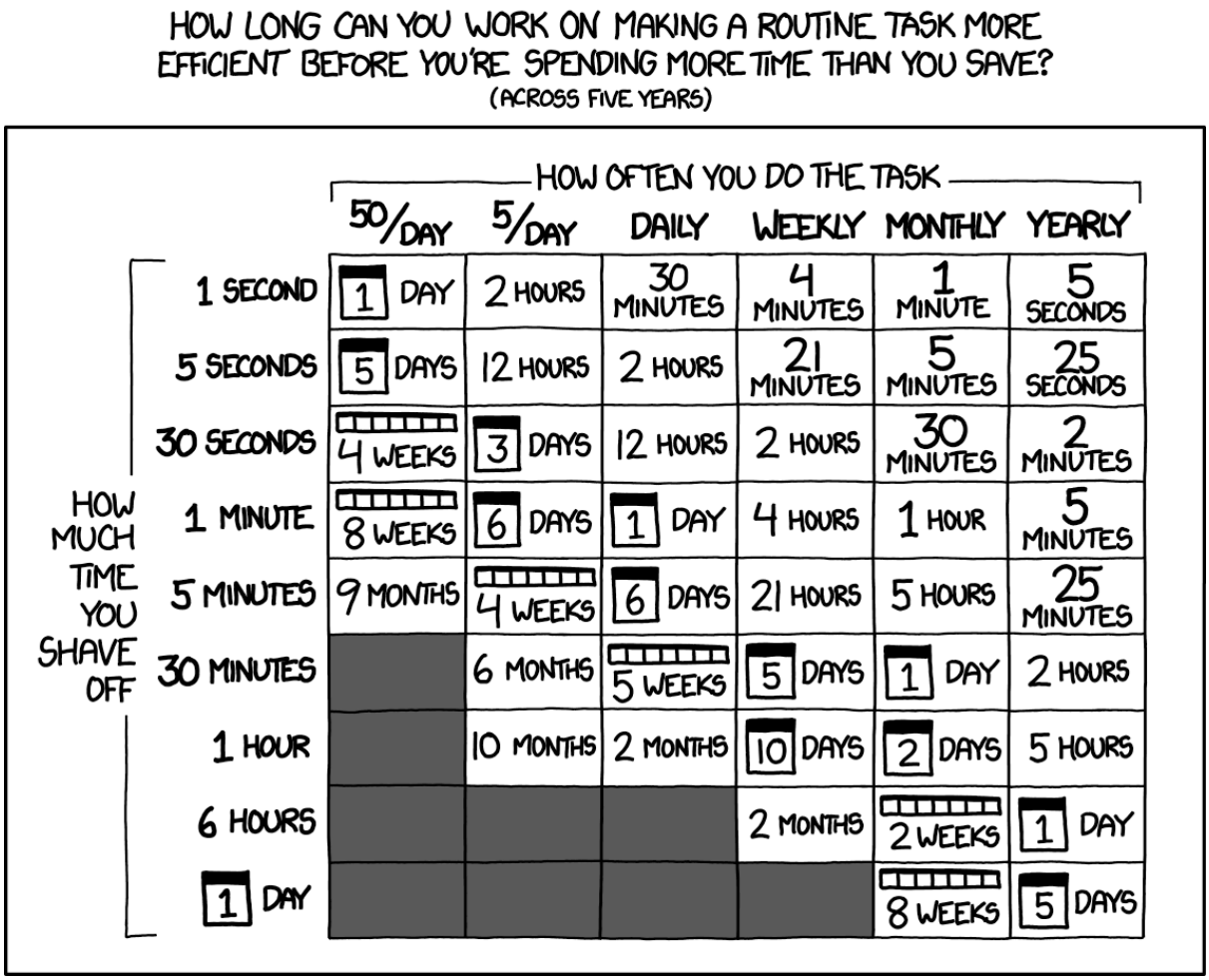

Figure 5: Automation Break-even Point