Definition of Done

Neronet

Toolbox for managing the training neural networks

CSE-C2610
Software Project

Aalto University

December 3, 2015

Aalto University
School of Science
and Technology

Outline

Overview

We have defined *Done* in three levels:

1. At the backlog item (BI) level
2. At the sprint level
3. At the project level

In addition, important quality attributes are specified to ensure fulfillment of nonfunctional requirements.

Quality attributes

High priority quality attributes:

- Usability (easy to learn by newbies)
- Reliability (no crashes or lost data)
- Extendability (can be customized)
- Performance (low overhead)

DoD at BI level

A backlog item is *Done* if

- unit test coverage $> 90\%$
- functional system test coverage $> 80\%$
- code confroms to guidelines
- code is commented and documented
- independent peer review has been completed

DoD at sprint level

A sprint is *Done* if

- All BIs in the sprint backlog are *Done*
- Increment is tested and reviewed
- Sprint goal is achieved

DoD at project level

A project is *Done* if

- All sprints are *Done*
- The project is documented, tested and reviewed
- The project results reflect the Product Vision
- The PO, coach and course staff are satisfied

Process Overview

Neronet

Toolbox for managing the training neural networks

CSE-C2610
Software Project

Aalto University

December 3, 2015

Aalto University
School of Science
and Technology

Outline

Outline

Sprints

S	Start	End	D	Sa	Te	Tu	Jo	Ju	Ii	Ma
0	19.10.	13.11.	25	50	35	35	35	35	35	35
1	13.11.	4.12.	21	30	33	33	33	33	33	33
2	4.12.	11.1.	38	30	33	33	33	33	33	33
3	11.1.	1.2.	21	15	33	33	33	33	33	33
4	1.2.	29.2.	28	15	33	33	33	33	33	33
5	29.2.	21.3.	21	15	33	33	33	33	33	33
6	21.3.	11.4.	21	20	25	25	25	25	25	25

Note

- S2 includes exams (7.-18.12.) and holidays (23.12.-1.1.)
- S4 includes exams (15.-19.2.)
- S6 includes exams (4.-9.4) and is reserved for mainly polishing & documenting for final review (11.-13.4.)

Events

Time	Event	Participants
30.10. 16-18	Project kickoff	team + PO
13.11. 15-17	Sprint 0 demo	team + Coach
16.11. 11-13	Sprint 1 planning	team + PO
04.12. 16-18	Progress review I	team + PO + Coach

All events, locations, agendas and other details are uptodate in Google Calendar.

Outline

Overview

Recurring events:

- Sprint planning
- Sprint review
- Sprint retrospective
- "Daily" scrums
- Teamwork sessions

Sprint planning

A sprint planning session is organized at the start of each sprint.

1. Before the session
   - the PO makes sure the product backlog contains an ordered list of items with a description and a number depicting business value
   - the team plays planning poker to define effort estimates (story points) for each BI
2. During it the team and the PO
   - briefly define the increment's purpose, the sprint goal
   - move BIs from the product backlog to the sprint backlog
3. After it, the team
   - chews the BIs into smaller tasks
   - assigns effort estimates on the tasks by planning poker
   - assigns a developer and a reviewer to each task

Sprint review

At the end of each sprint, we

- demonstrate the stories we were able to get *done*
- adapt the product backlog based on the results, if needed

Sprint retrospective

After the sprint review, we

- evaluate and rank teamwork practices
- discuss how teamwork could be improved
- remove/replace any bad practices
- plan implementation of new improvements
- give feedback to sprint team leader

Daily scrums

On Wednesdays and Fridays we have a scrum in which everyone quickly explains what

- they did since last Scrum
- problems they have encountered
- they plan to do before the next Scrum

Work plans are adjusted depending on input.

Teamwork sessions

Most weeks, we'll

- have a Scrum and a 6h session on Wednesdays
- have a Scrum and a 5h session on Fridays
- do some individual work remotely to cover up any missed sessions

Team sessions are mainly held in Maari.

Outline

Overview

Used practices and tools:

- Testing & quality assurance: DoD
- Communication: Email, Flowdock, Hangout/Skype, WhatsApp
- Backlog management: Agilefant
- Time tracking: Agilefant
- Version control: GitHub
- Collaboration: Floobits, ShareLaTeX, Google Drive
- Motivation: Team Spirit Recap

Quality assurance

We guarantee quality by making

- sure team members adhere to the DoD.
- each member responsible for the quality of the code he reviewed.
- the PO is responsible for the business value of sprint goals and BIs and for making sure the team understands them.

Communication

We use the following channels:

- Email - communication that involves the PO and Coach
- Flowdock - general forum for everyday discussion
- WhatsApp/Phone - urgent team communication
- Skype/Hangout - remote teamworking sessions

The Sprint team leader communicates with the PO and Coach.

Backlog management

Agilefant is used for all backlogs.

- Version 1 - the product backlog
- Sprint 0-6 - the sprint backlogs

Time tracking

We track our worktime with Agilefant. We log each work session duration to the story or task we worked on.

Version control

We use Git with GitHub with branches:

- stable - tested and working version
- sprint - increment work in progress
- story$X$ - story work in progress

Our development process has four steps:

1. We assign a developer and a reviewer for each story
2. The developer solves the story in a new branch
3. Then he asks the reviewer for a merge review
4. The reviewer determines whether the work meets the story requirements and the DoD
   - if not, he asks the developer to continue working on it
   - if yes, he merges the story branch to the sprint branch and the developer marks the story as *done*

Collaboration

When we work simultaneously on the same documents we use Floobits, ShareLaTeX, or Google Drive depending on the document.
Floobits is connected to a Git repo clone which facilitates when working with many files. It is particularly suitable for collaborative code level planning and code reviews.

Motivation

Team Spirit Recap:

- Mission: Why we exist
  - Create useful software for Pyry (and others)
  - We are doing this project to learn: software development, requirements engineering, architecture, project management, quality assurance, Scrum, communication with client
  - We want grade five, quality award
- Values: What we believe in and how we will behave
  - Superior quality
  - Self-development
  - Respect
  - Achievement
- Vision: What we want to be
  - We want to see ourselves as the best of the course teams
  - We want to win the Quality award!
  - We want to get grade 5+.
  - We want to get an awesome reference (GitHub repo) that we can market on our future job applications.

Aalto University
School of Science
and Technology

Product Vision

Neronet

Toolbox for managing the training neural networks

CSE-C2610
Software Project

Aalto University

December 3, 2015

Aalto University
School of Science
and Technology

Outline

Why – business view

Currently available state-of-the-art tools and systems for computational research could be improved.
Researchers are slowed down by lack of good easy tools for practical everyday difficulties like

Why – business view

Currently available state-of-the-art tools and systems for computational research could be improved.
Researchers are slowed down by lack of good easy tools for practical everyday difficulties like

- managing a queue and history of different experiments

Why – business view

Currently available state-of-the-art tools and systems for computational research could be improved.
Researchers are slowed down by lack of good easy tools for practical everyday difficulties like

- managing a queue and history of different experiments
- specifying several variations of experiments and running them

Why – business view

Currently available state-of-the-art tools and systems for computational research could be improved.
Researchers are slowed down by lack of good easy tools for practical everyday difficulties like

- managing a queue and history of different experiments
- specifying several variations of experiments and running them
- getting information about the computing environment

Why – business view

Currently available state-of-the-art tools and systems for computational research could be improved.
Researchers are slowed down by lack of good easy tools for practical everyday difficulties like

- managing a queue and history of different experiments
- specifying several variations of experiments and running them
- getting information about the computing environment
- monitoring and controlling progress of ongoing experiments

Why – business view

Currently available state-of-the-art tools and systems for computational research could be improved.
Researchers are slowed down by lack of good easy tools for practical everyday difficulties like

- managing a queue and history of different experiments
- specifying several variations of experiments and running them
- getting information about the computing environment
- monitoring and controlling progress of ongoing experiments
- analysing and comparing the results of experiment variations

Why – business view

Currently available state-of-the-art tools and systems for computational research could be improved.
Researchers are slowed down by lack of good easy tools for practical everyday difficulties like

- managing a queue and history of different experiments
- specifying several variations of experiments and running them
- getting information about the computing environment
- monitoring and controlling progress of ongoing experiments
- analysing and comparing the results of experiment variations

This leads to ineffective use of man and machine hours.

What – product goals

The product's goal is to enable easy

1. specification of experiments and management of queues
2. batch submission of experiment jobs to computing clusters

What – product goals

The product's goal is to enable easy

1. specification of experiments and management of queues
2. batch submission of experiment jobs to computing clusters
3. monitoring of ongoing experiments' logs and parameter values
4. access to experiment information during and after the run
5. configurable notifications on experiment state and progress
6. configurable criteria for experiment autotermination

What – product goals

The product's goal is to enable easy

1. specification of experiments and management of queues
2. batch submission of experiment jobs to computing clusters
3. monitoring of ongoing experiments' logs and parameter values
4. access to experiment information during and after the run
5. configurable notifications on experiment state and progress
6. configurable criteria for experiment autotermination
7. logging of experiment history
8. preferences configuration

What – product goals

The goals should be achieved in a generic way suitable for many different computational problem areas and experiment types.

What – product goals

The goals should be achieved in a generic way suitable for many different computational problem areas and experiment types.
Potential extra goals:

1. visualisation and analysis of experiment data
2. some neural networks specific functionality

What – product goals

The goals should be achieved in a generic way suitable for many different computational problem areas and experiment types.
Potential extra goals:

1. visualisation and analysis of experiment data
2. some neural networks specific functionality

Nonfunctional requirements:

1. low computational and memory overhead
2. good usability
3. easily maintainable and extensible
4. open source

For whom – users

The envisioned users are all individuals who run long lasting computational experiments and appreciate progress feedback.

For whom – users

The envisioned users are all individuals who run long lasting computational experiments and appreciate progress feedback. The potential user segments include for instance:

- Deep learning researchers
- Machine learning researchers
- Computational physics researchers
- Computational bioscience researchers
- Data science practitioners
- Enthusiasts & hobbyists

Progress review

Neronet

Toolbox for managing the training neural networks

CSE-C2610
Software Project

Aalto University

December 3, 2015

Aalto University
School of Science
and Technology

Outline

Introduction

Goals

Our goal is to develop a tool for computational researchers to enable easy

- specification and management of experiment queues

Introduction

Goals

Our goal is to develop a tool for computational researchers to enable easy

- specification and management of experiment queues
- batch submission of experiment jobs to computing clusters

Introduction

Goals

Our goal is to develop a tool for computational researchers to enable easy

- specification and management of experiment queues
- batch submission of experiment jobs to computing clusters
- monitoring of ongoing experiments' logs and parameter values

Introduction

Goals

Our goal is to develop a tool for computational researchers to enable easy

- specification and management of experiment queues
- batch submission of experiment jobs to computing clusters
- monitoring of ongoing experiments' logs and parameter values
- access to experiment information during and after the run

Introduction

Goals

Our goal is to develop a tool for computational researchers to enable easy

- specification and management of experiment queues
- batch submission of experiment jobs to computing clusters
- monitoring of ongoing experiments' logs and parameter values
- access to experiment information during and after the run
- configurable notifications on experiment state and progress

Introduction

Goals

Our goal is to develop a tool for computational researchers to enable easy

- specification and management of experiment queues
- batch submission of experiment jobs to computing clusters
- monitoring of ongoing experiments' logs and parameter values
- access to experiment information during and after the run
- configurable notifications on experiment state and progress
- configurable criteria for experiment autotermination

Introduction

Goals

Our goal is to develop a tool for computational researchers to enable easy

- specification and management of experiment queues
- batch submission of experiment jobs to computing clusters
- monitoring of ongoing experiments' logs and parameter values
- access to experiment information during and after the run
- configurable notifications on experiment state and progress
- configurable criteria for experiment autotermination
- logging of experiment history

Introduction

What

In essence the product is a Python-based tool that enables computational researchers to conduct their research more effectively.

Introduction

What

In essence the product is a Python-based tool that enables computational researchers to conduct their research more effectively.

- It utilizes SSH and TCP sockets to distribute the computational workload into computer clusters. It supports the Slurm job and resource manager but can function without it as well.

Introduction

What

In essence the product is a Python-based tool that enables computational researchers to conduct their research more effectively.

- It utilizes SSH and TCP sockets to distribute the computational workload into computer clusters. It supports the Slurm job and resource manager but can function without it as well.
- It is framework agnostic in that it permits the use of a very wide variety of tools to actually conduct the computing.

Results

Sprint 0

Goal: Team building and preparing for sprint 1

Results

Sprint 0

Goal: Team building and preparing for sprint 1 Done

Results

Sprint 0

Goal: Team building and preparing for sprint 1 Done
Product Backlog Items: *None*

Results

Sprint 0

Goal: Team building and preparing for sprint 1 Done
Product Backlog Items: *None*
Sprint 0 took a lot of effort from us since the project topic was very challenging to dive into. Also none of us had done this course before. Interviews with Jelena & Simo helped us to understand the project.

Results

Sprint 0

Goal: Team building and preparing for sprint 1 Done
Product Backlog Items: *None*
Sprint 0 took a lot of effort from us since the project topic was very challenging to dive into. Also none of us had done this course before. Interviews with Jelena & Simo helped us to understand the project.
We were proud of our efforts in the sprint.

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done
Product Backlog Items:

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI.

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources.

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources. Done

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources. Done
- US3: As a user, I want to specify experiments by name, files and parameters and edit and delete them.

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources. Done
- US3: As a user, I want to specify experiments by name, files and parameters and edit and delete them. Done

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources. Done
- US3: As a user, I want to specify experiments by name, files and parameters and edit and delete them. Done
- US4: As a user, I want to submit experiments to unmanaged nodes.

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources. Done
- US3: As a user, I want to specify experiments by name, files and parameters and edit and delete them. Done
- US4: As a user, I want to submit experiments to unmanaged nodes. Done

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources. Done
- US3: As a user, I want to specify experiments by name, files and parameters and edit and delete them. Done
- US4: As a user, I want to submit experiments to unmanaged nodes. Done
- US5: As a user, I want an experiment status report so that I can review experiment status details.

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources. Done
- US3: As a user, I want to specify experiments by name, files and parameters and edit and delete them. Done
- US4: As a user, I want to submit experiments to unmanaged nodes. Done
- US5: As a user, I want an experiment status report so that I can review experiment status details. Done

Results

Sprint 1

Goal: Develop a prototype that offers the most basic functionality via a CLI Done

Product Backlog Items:

- US1: As a user, I want a basic user guide that would cover the installation of Neronet and its use via CLI. Done
- US2: As a user, I want to specify clusters by address and type to specify my computing resources. Done
- US3: As a user, I want to specify experiments by name, files and parameters and edit and delete them. Done
- US4: As a user, I want to submit experiments to unmanaged nodes. Done
- US5: As a user, I want an experiment status report so that I can review experiment status details. Done

Just a prototype, a lot of work to do before user testing.

Demo

Demo script:

1. Presentation of Neronet's CLI user guide
2. Neronet Installation, preferences and initial setup of clusters
3. Specification of clusters via CLI
4. Specification of an experiment
5. Submission of the specified experiment to an unmanaged node
6. Retrieval of experiment status report
7. Experiment status report

Quality

Definition of done:

- We defined Done in three levels: BI, sprint and project
- BI level: unit test coverage 90%, functional test coverage 80%, conformity (PEP-8), commented, documented, peer reviewed
- Sprint level: BI:s are Done, increment is tested and reviewed, sprint goal is achieved
- We have followed our DoD almost as planned.

Quality

US	UTC	FTC	Com	Doc	Rev
1	-	0%	0%	0%	0%
2	0%	0%	0%	0%	0%
3	0%	0%	0%	0%	0%
4	0%	0%	0%	0%	0%
5	0%	0%	0%	0%	0%
	0%	0%	0%	0%	0%

Qualitatively we achieved our standards only partially:

- Unit and functional test coverage – only satisfactory
- Quality of comments and documentation – mediocre
- Peer review – done quickly

Quality

Used QA practices and tools:

- Python standard unittest framework
- Peer review

Relevant quality attributes:

- Usability
- Reliability
- Extendability
- Performance

Effort

S	Sa	Te	Tu	Jo	Ii	Ma
0	140/50	36/35	45/35	40/35	36/35	43/35
1	28/30	22/33	25/33	30/33	20/33	25/33
2	0/30	0/33	0/33	0/33	0/33	0/33
3	0/15	0/33	0/33	0/33	0/33	0/33
4	0/15	0/33	0/33	0/33	0/33	0/33
5	0/15	0/33	0/33	0/33	0/33	0/33
6	0/20	0/25	0/25	0/25	0/25	0/25
	168/175	58/225	70/225	70/225	56/225	68/225

Retros: Sprint 0

Sprint planning:

- backlog items must be clear and simple -teemu
- backlog items have been unclear, but the user guide probably helps
- it would have been better if the PO had created the stories from scratch -matias, tuomo
- the PO should give input when developing the user guide
- we should make sure we reserve enough time for the actual story selection on Monday -matias

Retros: Sprint 0

Daily scrums:

- we have mostly been doing teamwork, so there has been little new info in the scrums -Matias -Joona -Teemu
- they have been overly long and they have extended due to inexperience.
- people are late.

Retros: Sprint 0

Teamwork sessions:

- sessions are too long and sometimes people get hungry.
- generally someone has to leave early or comes late

Retros: Sprint 0

Tools:

- flowdock is good x6
- for remote work we have been using google hangout and skype. Skype has proven to be the most stable.
- for faster communication we are using whatsapp.
- agilefant has a steep learning curve. -Iiro
- people tend to forget to log their time at agilefant.
- hope to use more github during sprints
- floobits ain't very good. Doesn't seem to work in its intended purpose.
- Top 3 tools: 1) GitHub 2) Flowdock 3) Agilefant
- Worst 3 tools: 1) Floobits 2) Six tactics 3) Agilefant

Retros: Sprint 0

How teamwork could be improved:

- People should be more on time.
- hard to think on improvements on sprint 0

Technical Overview

Neronet

Toolbox for managing the training neural networks

CSE-C2610
Software Project

Aalto University

December 3, 2015

Aalto University
School of Science and Technology

Outline

Outline

Introduction

Neronet is a framework designed to facilitate the specification, submission, monitoring, control, analysis and management of many different computational experiments. The Neronet family consists of three members:

1. Neroman – a user run tool that acts as the *Neronet frontend* and user interface to provide access to Neronet's features and functionality. It is the researchers workhorse.
2. Neromum – a Nerokid manager daemon that helps the family stay organized while the kids are playing in distant clusters, possibly in collaboration with a Warden that is predesignated to supervise and control the playing grounds.
3. Nerokid – a tiny creature that looks after your computational toy and reports to mum in case it breaks.

Introduction

In more technical terms:

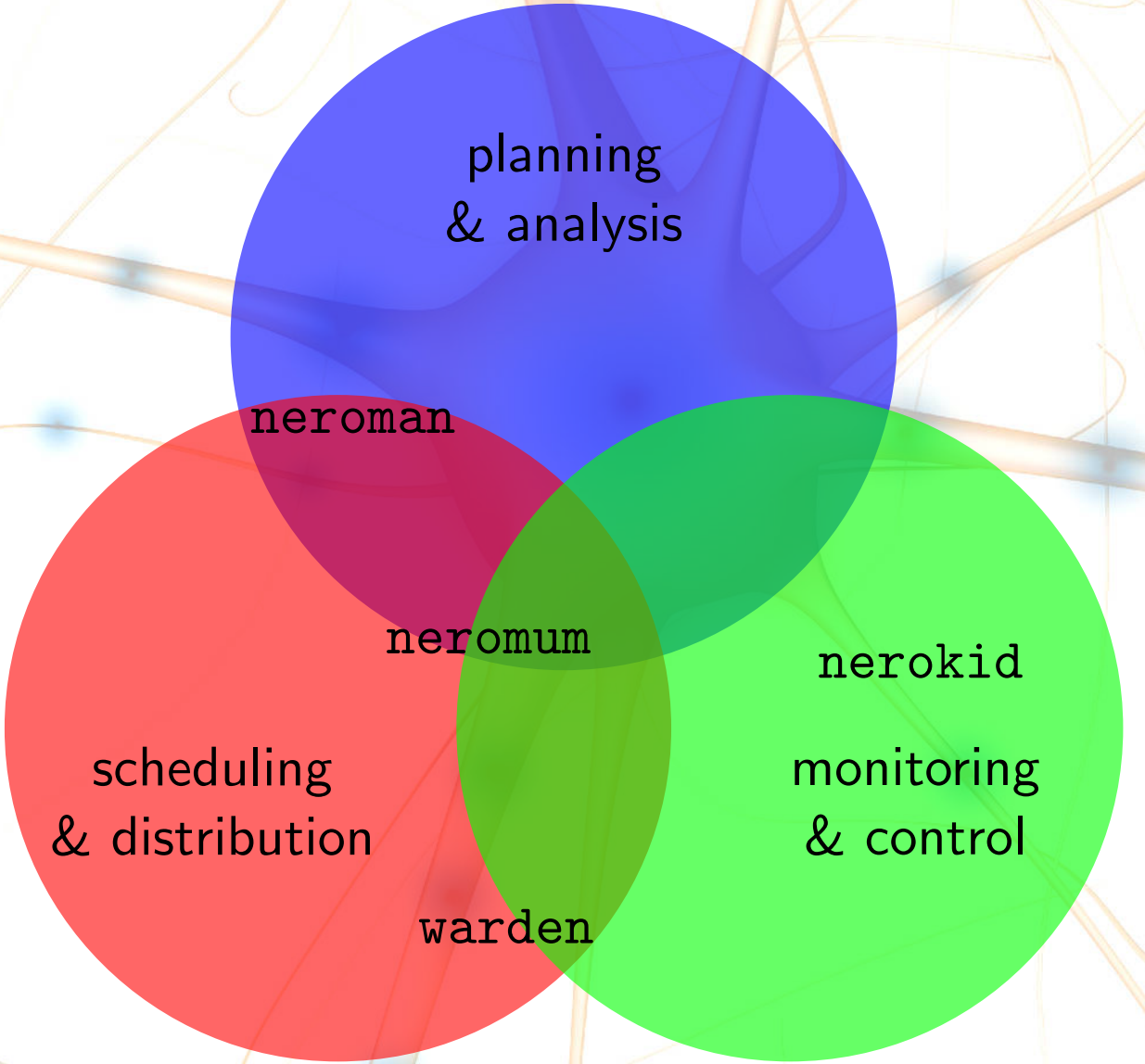
- Nerokid is a program designed to be launched by a job scheduler or Neromum directly in a cluster node to monitor, analyse and control a single experiment job and its environment in tandem with a Neromum
- Neromum is a job scheduler, manager and communications agent designed to work by herself or with a standard cluster job scheduler such as SLURM, OGE or Jobman (a Warden)
- Neroman is a tool that acts as the *Neronet frontend* and user interface to provide Neronet's functionality by dispatching Neromums in specified clusters or nodes with a bunch of jobs to do. The Neromums in turn dispatch the jobs to the Nerokids and then continue as communication intermediaries.

Introduction

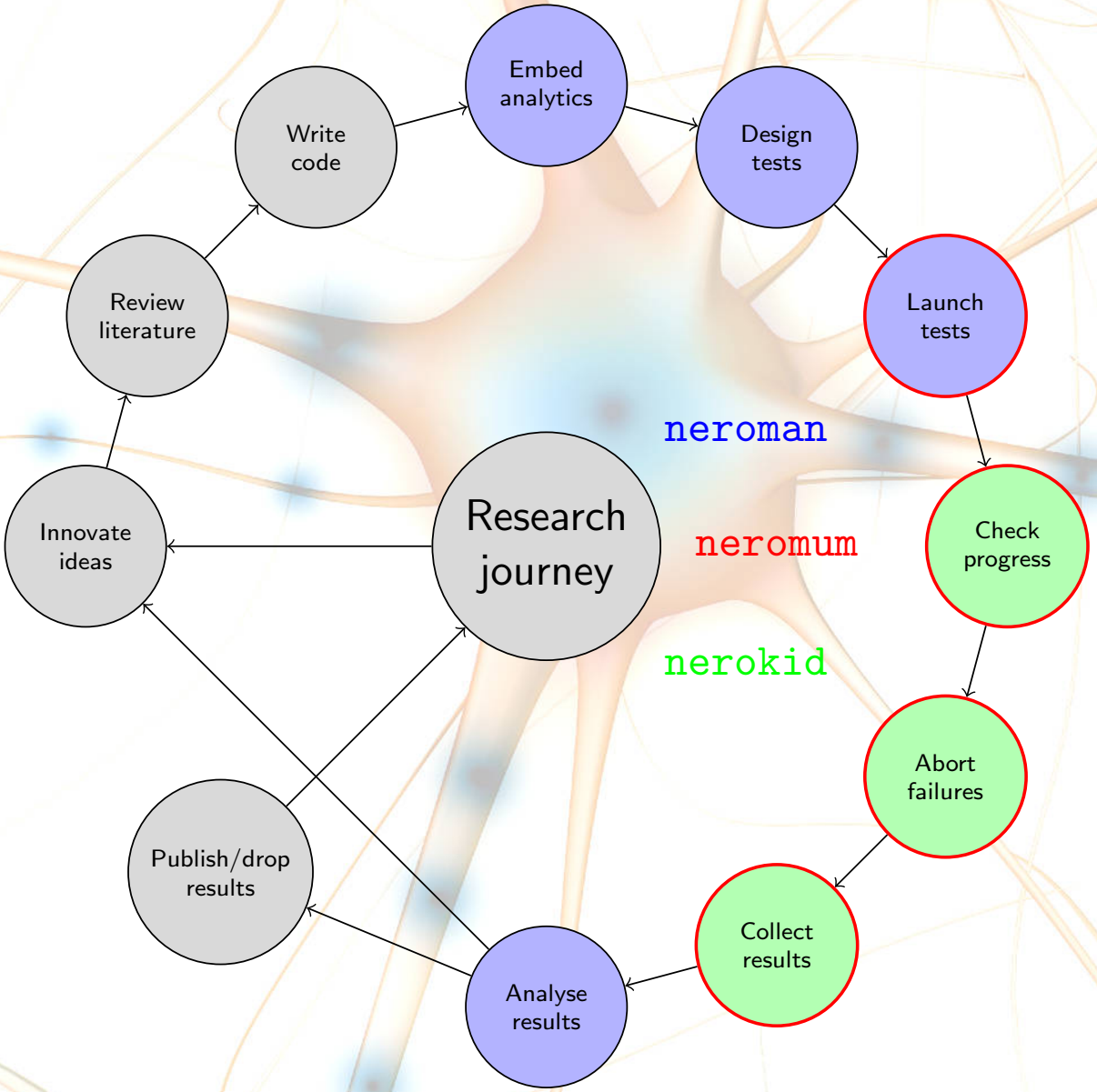
The following three slides provide a good introduction:

1. *Job management* – the associated problem domain spheres and how the system components are related to them
2. *Journey map* – the journey or work flow of a typical system user (a computational researcher) and the steps in which the Neronet tools are designed to be used
3. *Basic use case* – A sample basic system use case description

Job management



Journey map


Embed analytics
Write code
Design tests
Review literature
Launch tests
neroman
Research journey
neromum
Innovate ideas
Check progress
nerokid
Abort failures
Publish/drop results
Collect results
Analyse results

Basic use case

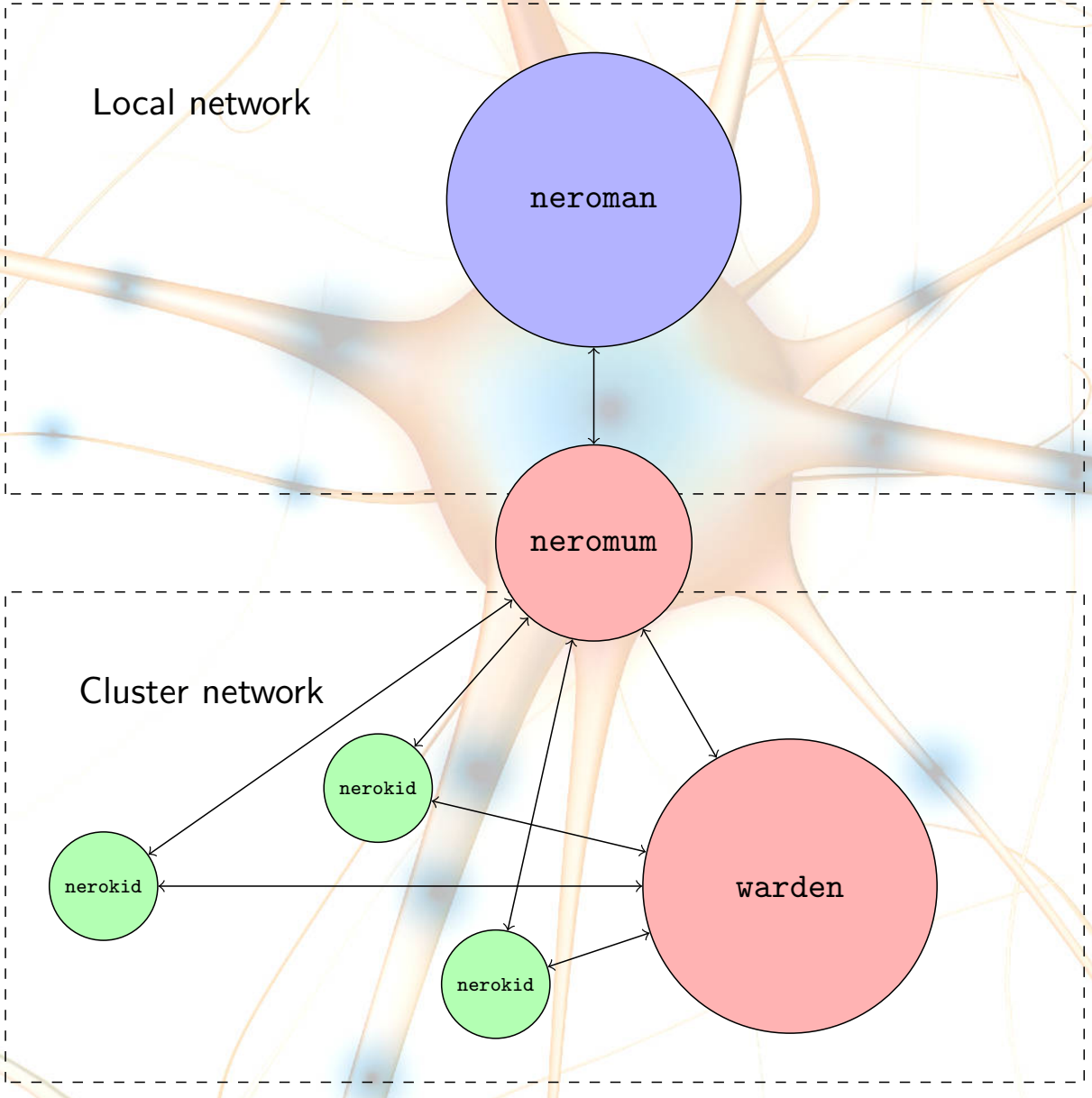
- **User**: A computational researcher
- **Goal**: To test how well a new design works with several different configuration options and parameter values
- **Preconditions**: SLURM cluster and Neronet setup, code and analytics developed and test inputs setup in Neronet compatible manner
- **Basic flow**:
  1. Specify a batch of Nerokids in the *config YAML* with parameters, inputs and other configurations
  2. Dispatch the jobs to the SLURM setup with autogenerated `sbatch` scripts and arguments: `neroman --submit triton 124-186`
  3. Receive and check progress notifications from email
  4. Monitor the experiment to see near realtime updates of analytics variable updates: `neroman --submit triton 124-186`
  5. Receive final results data and updates. To a log data file.
  6. Analyse, reiterate and/or publish results
- **Post conditions**: Computational experiments have been conducted in a very straightforward, effective and researcher friendly fashion

Outline

Components

- All Nero components are lightweight Python programs run with just the researcher's privileges on any modern *nix
- SSH is used for communication between Neroman and Neromums (user's existing ssh keys, ssh configs and privileges are used)
- Sockets are used between Neromums and Nerokids
- Neromum communicates with any Wardens using their CLIs and/or APIs
- The system is ment to be easy to setup, lightweight and the usability good for several types of uses.

Communication



neroman

- A daemon administered and configured by the researcher herself
- Should be run on a system with two way SSH access to the cluster gateway or nodes where the Neromums are deployed
- Key functionality
  - Facilitate and standardize experiment specification
  - Batch submit experiment jobs to Neromums
  - Send email notifications with progress data
  - Facilitate monitoring and control of running jobs
  - Autocollect key job results into a researcher specifiable format (f. ex. Excel)
  - Facilitate experiment analysis and history management
  - Lightweight and extendable with custom functionality
  - Configurable via YAML files

neromum

- A daemon started and administered by a Neroman.
- Should be run on a system with two way SSH access to the Neroman and socket access to the nodes where Nerokids are deployed
- Collaborates with standard Wardens.[1]
- Key functionality
  - Receive and execute commands from Neroman
  - Communicate with any Wardens by autogenerating job scripts (eg. `sbatch`) and their CLIs and/or APIs.
  - Collect information from Nerokids, process and cache them, send them to Neroman on request
  - Transmit commands from Neroman to the Nerokids

[1] SLURM is currently used by Triton and CSC. OGE is still used worldwide. Jobman could be setup for the CS gpu cluster.

nerokid

- A daemon started and administered by a Neromum directly or through a Warden on any modern *nix system (typically a cluster node) to start and monitor computational jobs
- Key functionality
  - Send information to Neroman via Neromum as configured
    - computing environment information
    - experiment job progress information read either through an API or by parsing output logs and data files (eg. CSV, JSON)
  - Interact with the Warden as specified (eg. autotermination based on poor experiment progress)
  - Lightweight and extendable with custom functionality
  - Configurable via YAML files and perhaps a Python API

Outline

Remarks

- A server (Neroman) per user approach is chosen because
  - easy minimalist setup (easy to try)
  - no need for special privileges
  - fully customizable by the user herself
  - low resources overhead per setup
  - total number of users relatively low as well
- SSH is used because
  - it is an existing standard among target system environments
  - user's existing SSH keys and configs provide an easy and effective way to provide secure networking
  - no need for network, port routing or privileges adjustments

Remarks

- Python 3 is used because
  - it is already available in most modern *nix systems
  - it has good support for the known software requirements
  - it is already popular among computational researchers
  - many related research libraries use it (Scipy, Numpy, Theano, Lasange, Pylearn, Blocks)
- Sockets are used because
  - they provide fast and efficient realtime communication in tightly connected networks
  - they are lightweight for the cluster filesystem