Which functions and concepts were contributed to the project by each team member:

general

			Coding							Refactoring/Documentation						
			WA	FP	JD	PJ	SY	MK	RL	WA	FP	JD	PJ	SY	MK	RL
github workflows				0,5	0,25		0,25									
run/kill script				1												
readme											0,8		0,2			
userguide											0,3	0,1	0,1	0,5		

webots

			Coding							Refactoring/Documentation						
			WA	FP	JD	PJ	SY	MK	RL	WA	FP	JD	PJ	SY	MK	RL
documentation										1						
controller	internal	internal_com			1					0,5		0,5				
		tcp			1					0,5		0,5				
		util	1							0,5		0,5				
		internal	0,8		0,2					0,5		0,5				
	supervisor	sv_com			1					0,5		0,5				
		sv_functions	1							0,5		0,5				
		tcp			1					0,5		0,5				
		util	1							0,5		0,5				
		supervisor	1							0,5		0,5				
worlds			1													
robots			1													

controller

			Coding							Refactoring/Documentation						
			WA	FP	JD	PJ	SY	MK	RL	WA	FP	JD	PJ	SY	MK	RL
documentation											0,5	0,5				
general	main			0,5	0,5						0,5	0,5				
	print			0,2	0,8						0,5	0,5				
	util			0,5	0,5						0,5	0,5				
	silhouette			0,5	0,5						0,5	0,5				
	Makefile			1	0						0,5	0,5				
webots	discr			0,5	0,5						0,5	0,5				
	drive			1	0						0,5	0,5				
	navi			1	0						0,5	0,5				
	pid			1	0						0,5	0,5				
	safe			0,2	0,8						0,5	0,5				
	tcp			0	1						0,5	0,5				
	wb_com			0	1						0,5	0,5				
	webot_worker			0,5	0,5						0,5	0,5				
backend	backend_com			1	0						0,5	0,5				
	backend_worker			0,5	0,5						0,5	0,5				
	udp			1	0						0,5	0,5				
tests	tests			0,5	0,5						0,5	0,5				

backend

			Coding							Refactoring/Documentation						
			WA	FP	JD	PJ	SY	MK	RL	WA	FP	JD	PJ	SY	MK	RL
General	ReadMe										0,3		0,3		0,4	
	UseMe notebooks					0,1	0,9							1		
Experiments: trainingsruns / notebooks	fakegym					0,25	0,25	0,25	0,25							
	webotsgym			0,33	0,33	0,33										
fakegym	action					1										1
	state					1										1
	fakeenv					0,8	0,025	0,15	0,025				0,3			0,7
	utils					0,7	0,3									1
webotsgym	com	automate		0,33	0,33	0,33							1			
		communicate		0,40	0,20	0,4							0,9		0,1	
		package		0,25	0,25	0,5							0,9		0,1	
		state				1							0,9		0,1	
	env	reward			0,1	0,5		0,4					0,5		0,5	
		action				1							0,6		0,4	
		observation				1							0,6		0,4	
		grid				0,8		0,2					0,6		0,4	
		reward				0,7		0,3								
		webotenv				1							0,8		0,2	
	utils	seeding				1							0,2		0,8	
		io				1							0,2		0,8	
		misc				1							0,2		0,8	
		plotting				0,1	0,9						0,2		0,8	
		seeding				1										
james				0,25		0,75							0,3		0,7	
tests							0,2	0,2	0,6						0,3	0,7
research	reward					0,15	0,1	0,65	0,1							
	gym					1										
	observation/action					0,8		0,2								
	algorithms						0,5		0,5							

What each team member has done to fulfill their respective management role:

Jonas Dommes (Integration Manager)

- Designed and implemented communication interfaces between webots (internal and supervisor), external controller and backend
    - Chose udp for external controller - backend communication, because only the newest data is relevant, if packages arrive later they can be discarded
    - Chose tcp for the other interfaces, because they would not exist in a real use case scenario (supervisor wouldn't exist at all, external controller and internal controller would be in the same device)
- Updated the interfaces with new values and new request-response behaviour (see backend_worker.c)
- Set up automated compiling for webots controllers
- Set up folder structure / notion database to log experimental training runs

Wladimir (Integration Manager)

- Research into interfacing to Webots using the Webots API for C
- Modification of .proto files to ensure expected results from sensors
- Building the Webots interface for both internal and supervisor controller
- Supervisor <-> backend communication protocol and parameter design
- Parameter checking for the supervisor (debug)
- Co-design of the „internal controller“ <-> „external controller“ communication
- Interface discussions to satisfy the requirements whilst respecting the limitations
- Overall maintenance/updates to ensure interoperability of the different levels

Per Joachims (Time Manager)

- Setup of Notion Kanban board and current task working table for the next sprint, frequent communication via Slack with all team members concerning the goals that were set and how the progress is
- Long term planning of project, e.g. use custom gym to use with stable baselines models:
    - FakeGym for faster deployment of WbtGym in mid-project-phase + conduct experiments in backend
    - Transition to WbtGym with transfer learning
    - Setup of WbtGym for automated training (grid and continuous world)
- Design of FakeGym and WbtGym:
    - Specification of configurations for the WbtGym
    - Definition of requirements for Webots/external controller for normal communication protocol and automated training. Collaborative work on packet transmission between external controller and backend. Setup of manual controllable backend (james.py) for test runs in run.sh-script
    - Coordination of training runs and evaluation of different models in the FakeGym and WbtGym

    - Setup and distribution of work packages for research in the backend, e.g. important hyper parameters, model explanation, reward tweaks, behaviour cloning, pathfinding

Fabian Peddinghaus (Technical Manager)

- Setup version control concept / repository workflow, choose version control system (Git) and setup hosting service (Github)
- Implement basic branch, merge and directory architecture and introduce team members to architecture
- Setup main communication method (slack) and implement channels
- Help / assist with merge conflicts and other infrastructure related issues
- Start and define coding guideline
- Work together with integration manager to define and implement platform (posix) dependent communication interface
- Provide makefile, scripts and readme to compile, start and stop code stack (hole simulation)
- Search for CI tool (first choice was TravisCI, but switched over to Github actions early on)
- Implement and connect CI with Git, readme and unit tests (automatic compilation and testing using tests)
- Monitor and maintain Git and CI infrastructure
- Assists team members with implementation of unit tests and linting
- Setup integration tests (not successful due to limitations of webots)

Mats Kipper (Quality Manager):

- Research on general approach to solve fake and webots environment
- Research for the creation of an own DQN algorithm
- Research and testing of different reward functions, observation spaces and action spaces on fake environment
- Testing of DQN algorithm (stable_baseline) on fake environment
- Testing of PPO1 in discrete and continuous action space
- Implementation of code review cycles
- Implementation of integration tests
- Documentation of webots and readme’s

Shanshan Yin (Quality Manager):

- Check python code periodically and keep communication with other members to fix the bugs. Keep the code in high score in Pylint.
- Setup workflow for linter and unit tests.
- Implementation of part of unit tests.
- Research on different reinforcement learning algorithms and reward functions.
- Testing of PPO algorithm in the discrete world (observation, reward functions, training parameters,etc.).

- Implementation of the usage examples in /UseMe.
- Compilation of UseGuide.

Rui Li (Quality Manager)

- Research on different RL algorithms to solve discrete and continuous action space & explanation for comparison of DQN and PPO1
- Research on tips for improving reward function
- Testing of different algorithms and different reward functions on fake environment
- Training and Testing of different models with PPO1 on fake environment
- Implement the basic method for Pre-training (behavior cloning) on continuous action space in webot environment
- Implement the unit tests for fake environment
- Implement an environment check function for webot environment
- Refactoring and documentation of fake environment