**Architecture Of Smart Factory ICS**

**(Turtle Intelligence developer edition)**

**Erik Palenčík**

Table of Contents

[1. Intro 4](#_Toc199684685)

[1. About the author 4](#_Toc199684686)

[2. Motivation to write the book 4](#_Toc199684687)

[3. Motivation to create Turtle Engine 4](#_Toc199684688)

[4. Where to find project 5](#_Toc199684689)

[2. Simulation theory 6](#_Toc199684690)

[5. Monte Carlo method 6](#_Toc199684691)

[6. Probability distribution 6](#_Toc199684692)

[7. Statistics 6](#_Toc199684693)

[3. Heuristics 7](#_Toc199684694)

[8. Linear programming 7](#_Toc199684695)

[9. Genetics algorithm 7](#_Toc199684696)

[4. Data structures and algorithms 8](#_Toc199684697)

[10. Queue 8](#_Toc199684698)

[11. Buffer 8](#_Toc199684699)

[12. Maps 8](#_Toc199684700)

[13. Djikstra 8](#_Toc199684701)

[5. Entities and definitions 9](#_Toc199684702)

[14. Basic entities 9](#_Toc199684703)

[15. Smart entities 9](#_Toc199684704)

[6. Platform architecture overview 10](#_Toc199684705)

[16. Entities and actors explanation 10](#_Toc199684706)

[17. Simulation initialization 10](#_Toc199684707)

[7. Statistics and analysis 12](#_Toc199684708)

[8. Working with turtle 13](#_Toc199684709)

[18. Application installation 13](#_Toc199684710)

[19. Basic of the application 13](#_Toc199684711)

[9. Simulation software problems and challenges 14](#_Toc199684712)

[20. Dependency tree 14](#_Toc199684713)

[21. Root problem of dependency tree 14](#_Toc199684714)

[10. Platform configuration and plugins 16](#_Toc199684715)

[11. Used literature 17](#_Toc199684716)

[12. Licenses and project dependencies 18](#_Toc199684717)

[22. Frontend 18](#_Toc199684718)

[23. Backend 18](#_Toc199684719)

# Intro

In modern companies the word simulation is necessity:

What can simulation software help us:

* Verify our idea
* Show weak spots
* E-documentation

## About the author

Hello, my name is Erik Palenčík, while writing this book I work as Senior Software developer and starting my PhD study on the faculty of the mechanical engineering in Žilina – Industry Engineering field.

During study of IT (which I failed) I continued in this field by work, but I finished bachelor’s in management and master’s degree in Industry Engineering. I have now 9 years of experience with creation of 3D applications on web or desktop, and multiple experiences with creation of VR simulation of company processes for training purposes. Also last few years I actively develop digital twin platform for monitoring, analysis and optimalizations.

## Motivation to write the book

As student I went to multiple books – maybe great in simulations community, but for me as free time simulation worker, they were useless, after reading four hundred pages, I read no information. So, this was the reason why to create my own publication to help students and bring fun to the simulations world.

## Motivation to create Turtle Engine

During my work I started to experiment with Three JS and web and backend technologies, as my skillset grown and my life passed through, I decided to create my own digitalization platform – for uploading 3D models, point clouds, 360 panoramas and another interactive content.

By the time I undergone Modeling and Simulations and learning about manual simulation in this course I decided to create my own simulation engine with strong 3D visualization and knowledge of web technologies. Also, terrible UI and performance of another simulation tools like Anylogic (where UI performs terrible even on high end PC) and strong licensing like in Tecnomatix Plant simulation I decided to create my own platform, which should be modern, open source, and deployable everywhere.

Another crucial motivation was to create my own, where I will be author, and I will be free to share source code to every company or person I will be ever in contact with. There was also motivation in moving forward my software architecture skills – you can read a million books, but you will **NEVER** move forward until you sit down and try it yourself.

Turtle Engine is part of my free time activities and my university research. Platform also contains modules to boost learning and visualization of students, not only to make practical examples.

## Ultimate goal of the book

As part of my PhD study, my research goal was Intelligent logistics factory of 21st century. As someone moving in manufacturing world and IT world this task looked like impossible. The main reason is that companies today don’t have IT problems, they have more personal, demand or cost problems – with rising prices of electricity.

As the result of creation of me should be the last chapter with name **“Divide and conquer”**. The chapter will describe architecture (and also trial) to create ƒultimate corporate solution, which can be probably the ultimate AI project of life.

## Where to find project

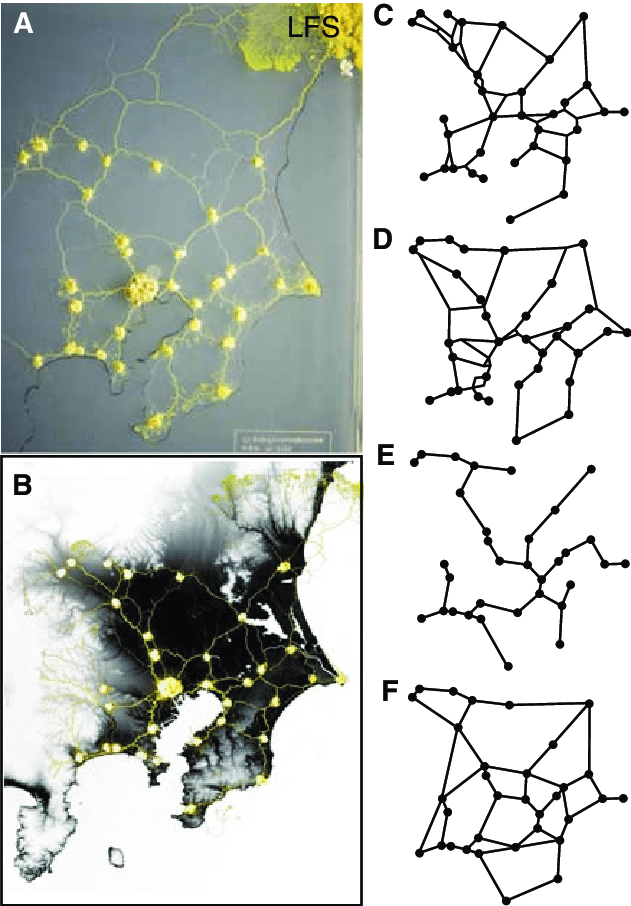
<https://github.com/erikpa1/GameOfResources_Backend>

<https://github.com/erikpa1/GameOfResources_Frontend>

<https://github.com/erikpa1/TurtleSim>

# AOF – Artificial organic factory

Goal of my PhD study contained also study of living organism and their implementation in hard technical environment. As an inspiration for this idea was research of Japanese research about potential of bio organisms to calculate the most optimal paths for metro.



Source

<https://www.researchgate.net/figure/Comparison-of-the-Physarum-networks-with-the-Tokyo-rail-network-A-In-the-absence-of_fig2_41111573>

As another point to this idea was the way how human neural network is created, the way how pain, struggle and repetition of task strengthens neural network connections. Why can’t we achieve something like this in cold hardware world? The neural theory says, that you can’t just replace human brain, because neural networks are connected for the body we are born in, so our brain can’t work in different body.

As the result of this idea was, why we could not software which would spread across company server – like living biome and optimize itself to perfect state.

A diagram of a company

Description automatically generated

Figure Divide and conquer architecture

As we can compare the following two pictures, there is no difference. Every node in in biome contains some data and resources (we can call them requirements).

A diagram of a company

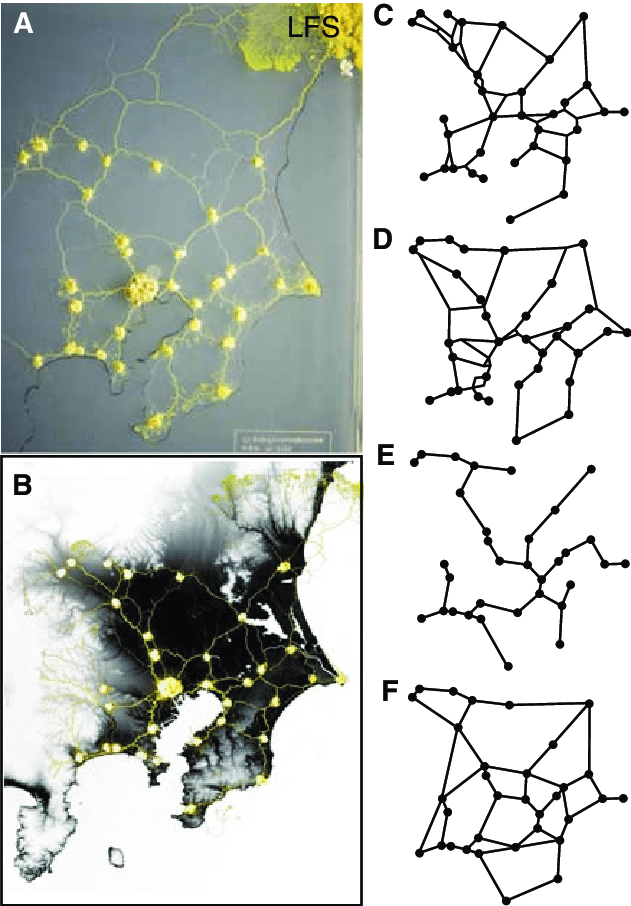
Description automatically generated

Figure Agentic system and biological computer

# Probability distribution

# Simulation theory

## Monte Carlo method

Monte Carlo is the most popular simulation method

## Probability distribution

## Statistics

# Heuristics

## Linear programming

## Genetics algorithm

# Data structures and algorithms

## Queue

## Buffer

## Maps

## Djikstra

# Entities and definitions

## Basic entities

The purpose of basic entities is to create simplified simulation with basic set of basic behavior.

**Spawn**

Spawn is behavior responsible for creation of entities. Spawn can work in the following modes:

1. Constant spawn time – spawn every 5 seconds
2. Normal distribution spawning – spawn in NormDist(10, 30)

**Process**

**Delay**

**Buffer**

**Queue**

**Merge**

**Split**

**Switch**

**Sink**

An opposite to spawn, sink destroys entities, sink is responsible for removing entity from the world.

## Smart entities

The purpose of smart entities is to add additional value to

# Platform architecture overview

Many computer languages support different architecture allowing different patterns and code architecture. Because Turtle Sim was developed with usage of Golang (as backend) – functional static typed language and JavaScript (frontend – React and Three JS) - duck typing OOP language, there are special kinds of architectures used in these cases.

## Entities and actors explanation

In Turtle we differentiate **Entities** and **Actors** – in IT theory they can be the same thing, but in this cases I decided to use this two names separately.

Entities are objects in world which has some role and behavior, e.g. Process, Spawn, Sink, …

Actors are objects traveling between entities being able to change it’s state, wear states.

As multiple software Turtle is based on Entity which can have a certain behavior. Entity can be for example:

* Process behavior
* Queue behavior
* Sink behavior
* …. another

Those behavior have something in common – they can provide or take actors. For handling different combination functions sim behaviors have to implement set of interfaces to be able. Interfaces entity can implement are:

* ISimBehavior – this is basic interface every entity must implement.
* ActorTakerBehavior – this interface mean that entity can take actor
* ActorProviderBehavior – this interface mean that you can ask entity for some actor if available.

A diagram of a diagram

Description automatically generated

## Simulation initialization

As every system also Turtle models have to somehow prepare load data, prepare them and then simulate.

A diagram of a process

Description automatically generated

Figure Simulation Initialization

The following loading stages are:

1. **World deserialization** – in this phase world is loaded from database.
2. **Init 1** – in this phase there is called function **Init1** in which you entity should prepare data

## Upcoming events

To optimize data flow (and performance of app) in the communication there is mechanism called Upcoming Events. Upcoming events collects times of state change for every entity and sends it to frontend which is responsible to run the timers to achieve this goal.

A screenshot of a computer

Description automatically generated

Figure Upcoming events

Upcoming event can have following states:

* Going to **SPAWN**
* Going to **UNSPAWN**
* Going to **FINISH**
* Going to **CANCEL**

# Statistics and analysis

## Bottle neck analysis

## Charing capacity

# Implementation of LLM and agentic compatible architecture

As part of my research I had to interact somehow with LLM models.

# Implementation of neural network predictions

# Implementation of reinforcement learning to an existing application

# Working with turtle

## Application installation

Unfortunately comparing with another simulation platforms. Turtle is cloud-based application built on MongoDB, not file structure, where files can be easily moved and stored on PC.

For Turtle at first you have to Install MongoDB (community version is enough). Actual application runs in version 7-8, never tested the newer one.

<https://www.mongodb.com/try/download/community-kubernetes-operator>

After installing MongoDB you can run pre-built executable file from following link “xxx”. Please select the platform you are currently on. Application uses in MongoDB database with name “turtle”, so please if this is inefficient use environment variable “TURTLE\_SIM\_DB\_NAME”.

## Basic of the application

After opening simulation platform, we can find similar interface being seen in environments like Anylogic or Tecnomatix Plant simulation.

A screen shot of a computer

Description automatically generated

# Simulation software problems and challenges

This chapter is mainly sum up problems and drawbacks I had to challenge during the creation of software.

## Dependency tree

As the platform simulation time is made in seconds (even milliseconds would have this problem), there comes one problem:

“When all process entities finish in the same step seconds how they will move actor next?”

This problem is not visible (or it’s error is so low on long run that it is irrelevant).

Solution – **Dependency tree**

**XXX – popisat hierarchiu dependency tree**

As solution looks promising it’s not helping. In following problem we will find why this problem is not an solution to our problem.

## Root problem of dependency tree

As you can see on the image bellow. We came to the root of dependency tree, but we see that flow can start from two places. Red spawn point spawns entities in 00:03 second interval and the blue one in 00:05. Station afterwards manufactures in 00:10 seconds process time. Problem here is that because of **red** spawner is first in step tree, the will be always first so blue one, will be always blocked.

A diagram of a network

Description automatically generated with medium confidence

Figure Root problem of dependency tree

The idea I came with is creation of lock mechanism, where blocked entities write their request to order unblocking status and this unblocking status should be primary order rule for entities sorting.

# Divide and conquer

The idea I created is to use LLM for the same principle. We can achieve this solution by following steps as user:

1. Define company systems

## Divine and conquer architecture advantages

The advantage of following system is similar as real human neural network – finding the most optimal way from point A to point B when firing electric signals.

# Platform configuration and plugins

Platform uses following environment variables:

|  |  |  |
| --- | --- | --- |
| Variable name | Default value | Explanation |
| TURTLE\_SIM\_DB\_NAME | turtle | This is key name used in query for database name |
| TURTLE\_SIM\_DB\_CONN\_STR | mongo://localhost:27107 | connection string to local MongoDB database |

# Used literature

# Licenses and project dependencies

All resources used to create this software are free and under MIT license. I would especially thanks to authors of:

## Frontend

* ThreeJS - incredible 3D rendering platform which enabled creation of this project.
* @react – fiber – thanks gentlemen for binding three JS to react, you created incredible architecture and developer friendly experience. Without it this project would be pain in ass and probably would fail.
* @react-drei – guys, big love to you, without you and your ideas my software would not be so nice and beautiful
* Antd design – thanks for providing this beautiful UI library for free.

## Backend

* Golang – best thing which was created by google, I fell in love with this language after a few lines of code
* Gin Gonic – amazing web server framework thanks for it creation.