REX: A DEVELOPMENT PLATFORM AND ONLINE LEARNING APPROACH FOR RUNTIME EMERGENT SOFTWARE SYSTEMS

Barry Porter, Matthew Grieves, Roberto Rodrigues Filho and David Leslie

School of Computing and Communications
Department of Mathematics and Statistics
Lancaster University, UK

b.f.porter@lancaster.ac.uk
Motivation

• Modern software remains **highly complex** to design, implement, maintain & configure, particularly for dynamic environments—this causes **high development costs** and **under-performing code**

• The state of the art in solving this is **self-adaptive systems**, which exhibit some awareness of **self** and of **environment**—However, these approaches relate only to the **configuration** element of systems development, and are also very **manual** in their definition, by:
  - Designing the base system as a non-adaptive one
  - Deciding which points of that system should be adaptable
  - Writing rules to determine how and when adaptation happens
**CONCEPT OVERVIEW**

- We propose a paradigm of **continuous self-assembly**, in which the *initial construction* of software and its *later adaptation* are one continuous machine-driven process.
CONCEPT OVERVIEW

• We propose a paradigm of continuous self-assembly, in which the initial construction of software and its later adaptation are one continuous machine-driven process.

Goal
(description)
(unit tests)

Start with a goal and a pool of behavior fragments.
CONCEPT OVERVIEW

• We propose a paradigm of **continuous self-assembly**, in which the initial construction of software and its later adaptation are one continuous machine-driven process.

Start with a goal and a pool of behavior fragments

Form a working system for the goal, deploy in production environment and monitor **performance and environment**
We propose a paradigm of **continuous self-assembly**, in which the *initial construction* of software and its *later adaptation* are one continuous machine-driven process.

- Start with a goal and a pool of behavior fragments
- Form a working system for the goal, deploy in production environment and monitor performance and environment
- Learn & find (or create) variants to better shape the system to its environment

**Goal**

(description) (unit tests)

**Goal**

(description) (unit tests)

**Goal**

(description) (unit tests)
CONTRIBUTIONS

• **Implementation platform (Dana):** A programming language with which to create small software building blocks that can be assembled into emergent systems, with near-zero-cost runtime adaptation for online exploration.

• **Perception, assembly and learning framework (PAL):** A framework built with Dana to discover & assemble emergent software, perceive its effectiveness and deployment conditions, and feed perception data to online learning.

• **Online learning approach:** An application of statistical linear bandits, using Thompson sampling, to help solve the search space explosion inherent in our approach, by sharing beliefs about components across possible configurations.

*Example system: an emergent, self-assembling web server*
APPROACH IN DETAIL

RE\textsuperscript{X}

API
setMain | getConfigs | setConfig | getPerception

PAL

learning module

assembly module

perception module

Implementation platform

Component pool

Motivation
Overview
Contributions
Approach in Detail
Evaluation

[8/41]
**Approach in Detail — Implementation Platform**

- Uses a *component-based development* paradigm, but:
  1. infused in a generalised systems programming language;
  2. supporting very fast, fine-grained runtime adaptation; and
  3. removing the need for wiring diagrams / configurations.
APPROACH IN DETAIL – IMPLEMENTATION PLATFORM

• Uses a **component-based development paradigm**

```java
interface RequestHandler {
    void handleRequest(TCPIdentifier s)
}

component provides App requires net.TCPIdentifier, net.TCPIdentifier, request.RequestHandler rh {
    int App::main(AppParam params[]) {
        TCPIdentifier host = new TCPIdentifier()
        host.bind(TCPIdentifier.ANY_ADDRESS, 8080)
        while (true) {
            TCPIdentifier client = new TCPIdentifier()
            if (client.accept(host))
                rh.handleRequest(client)
        }
        return 0
    }
}
```
APPROACH IN DETAIL — IMPLEMENTATION PLATFORM

- Uses a **component-based development** paradigm

```java
interface RequestHandler {
    void handleRequest(TCPSocket s)
}

component provides App requires net.TCPSocket, net.TCPServerSocket, request.RequestHandler rh {
    int App::main(AppParam params[]) {
        TCPServerSocket host = new TCPServerSocket()
        host.bind(TCPServerSocket::ANY_ADDRESS, 8080)
        while (true) {
            TCPSocket client = new TCPSocket()
            if (client.accept(host))
                rh.handleRequest(client)
        }
        return 0
    }
}
```
**APPROACH IN DETAIL — IMPLEMENTATION PLATFORM**

- Uses a **component-based development** paradigm

```java
interface RequestHandler {
    void handleRequest(TCPSocket s)
}

component provides App requires net.TCPSocket,
net.TCPServerSocket,
request.RequestHandler rh {
    int App::main(AppParam params[]) {
        TCPServerSocket host = new TCPServerSocket()
        host.bind(TCPServerSocket.ANY_ADDRESS, 8080)

        while (true) {
            TCPSocket client = new TCPSocket()
            if (client.accept(host))
                rh.handleRequest(client)
        }
        return 0
    }
}
```

Implementation variants: components that provide the same interface but implement it in different ways, yielding different behaviour or performance
APPROACH IN DETAIL

Motivation
Overview
Contributions

Approach in Detail

RE\textsuperscript{x}

PAL

API

setMain | getConfigs | setConfig | getPerception

assembly module
perception module

implementation platform

component pool

Dana

load | unload | getInterfaces | connect | adapt
A way to abstract entire software systems for machine learning into: reward; environment; actions

API

setMain | getConfigs | setConfig | getPerception

assembly module

perception module

load | unload | getInterfaces | connect | adapt
A way to **abstract entire software systems** for machine learning into: reward; environment; actions
**APPROACH IN DETAIL — PERCEPTION, ASSEMBLY AND LEARNING**

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions
A way to **abstract entire software systems** for machine learning into: reward; environment; actions

**Motivation**

**Overview**

**Contributions**

**Approach in Detail**

**Evaluation**
A way to **abstract entire software systems** for machine learning into: reward; environment; actions

**TABLE**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC</td>
<td></td>
</tr>
<tr>
<td>PT</td>
<td></td>
</tr>
<tr>
<td>Plain</td>
<td></td>
</tr>
<tr>
<td>Compressing</td>
<td></td>
</tr>
<tr>
<td>Com+Cache</td>
<td></td>
</tr>
<tr>
<td>Caching</td>
<td></td>
</tr>
<tr>
<td>GZip</td>
<td></td>
</tr>
<tr>
<td>ZLib</td>
<td></td>
</tr>
<tr>
<td>LFU</td>
<td></td>
</tr>
<tr>
<td>LRU</td>
<td></td>
</tr>
<tr>
<td>MFU</td>
<td></td>
</tr>
<tr>
<td>MRU</td>
<td></td>
</tr>
<tr>
<td>setMain</td>
<td></td>
</tr>
<tr>
<td>getConfigs</td>
<td></td>
</tr>
<tr>
<td>setConfig</td>
<td></td>
</tr>
<tr>
<td>getPerception</td>
<td></td>
</tr>
</tbody>
</table>

**API**

- load
- unload
- getInterfaces
- connect
- adapt
**APPROACH IN DETAIL — PERCEPTION, ASSEMBLY AND LEARNING**

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions

```
awk.bf.porter@lancaster.ac.uk

Motivation | Overview | Contributions | Approach in Detail | Evaluation
[19/41]

[WebServer.o,RequestHandlerTPC.o,HTTPHandler.o,TCP.o,File.o,...]
[WebServer.o,RequestHandlerTPC.o,HTTPHandlerZ.o,GZip.o,TCP.o,...]
[WebServer.o,RequestHandlerTPC.o,HTTPHandlerZ.o,CacheMFU.o,GZip.o,...]
[WebServer.o,RequestHandlerTPC.o,HTTPHandlerZ.o,CacheLRU.o,GZip.o,...]
[WebServer.o,RequestHandlerTPC.o,HTTPHandlerC.o,CacheMFU.o,TCP.o,...]
[...]
```
• A way to **abstract entire software systems** for machine learning into: reward; environment; actions

```
[WebServer.o, RequestHandlerTPC.o, HTTPHandler.o, TCP.o, File.o,...]
[WebServer.o, RequestHandlerTPC.o, HTTPHandlerZ.o, GZip.o, TCP.o,...]
[WebServer.o, RequestHandlerTPC.o, HTTPHandlerCZ.o, CacheMFU.o, GZip.o,...]
[WebServer.o, RequestHandlerTPC.o, HTTPHandlerCZ.o, CacheLRU.o, GZip.o,...]
[WebServer.o, RequestHandlerTPC.o, HTTPHandlerC.o, CacheMFU.o, TCP.o,...]
[...]
```
A way to abstract entire software systems for machine learning into: reward; environment; actions

- WebServer.o, RequestHandlerTPC.o, HTTPHandler.o, TCP.o, File.o, ...
- WebServer.o, RequestHandlerTPC.o, HTTPHandlerZ.o, GZip.o, TCP.o, ...
- WebServer.o, RequestHandlerTPC.o, HTTPHandlerCZ.o, CacheMFU.o, GZip.o, ...
- WebServer.o, RequestHandlerTPC.o, HTTPHandlerCZ.o, CacheLRU.o, GZip.o, ...
- WebServer.o, RequestHandlerTPC.o, HTTPHandlerC.o, CacheMFU.o, TCP.o, ...
- ...
**APPROACH IN DETAIL — PERCEPTION, ASSEMBLY AND LEARNING**

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions
APPROACH IN DETAIL — PERCEPTION, ASSEMBLY AND LEARNING

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions

```
interface Recorder {
    void addEvent(char name[], int value);
    void addMetric(char name[], int value, bool preferHigh);
}
```
**Approach in Detail — Perception, Assembly and Learning**

- A way to **abstract entire software systems** for machine learning into: reward; environment; actions
APPROACH IN DETAIL
An online learning algorithm which helps solve the search space explosion, and balances exploration and exploitation.
**APPROACH IN DETAIL — LEARNING MODULE**

- An online learning algorithm which helps solve the *search space explosion*, and balances *exploration* and *exploitation*.

  - Remove components with no variations.
  - Make regression factors for variants of a component above a baseline variant.

---

**Motivation**

**Overview**

**Contributions**

**Approach in Detail**

**Evaluation**
**APPRAOCH IN DETAIL — LEARNING MODULE**

• An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

> remove components with no variations
> make regression factors for variants of a component above a baseline variant
- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*
**APPROACH IN DETAIL — LEARNING MODULE**

- An **online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*

<table>
<thead>
<tr>
<th>( \beta_0 )</th>
<th>TPC.Plain</th>
<th>PT.Plain</th>
<th>TPC.Com.GZip</th>
<th>TPC.Com.ZLib</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_1 )</td>
<td>( \beta_0 1 )</td>
<td>( \beta_0 1 )</td>
<td>( \beta_0 1 )</td>
<td>( \beta_0 1 )</td>
<td>...</td>
</tr>
<tr>
<td>( \beta_2 )</td>
<td>( \beta_1 0 )</td>
<td>( \beta_1 0 )</td>
<td>( \beta_1 0 )</td>
<td>( \beta_1 0 )</td>
<td>...</td>
</tr>
<tr>
<td>( \beta_3 )</td>
<td>( \beta_2 0 )</td>
<td>( \beta_2 1 )</td>
<td>( \beta_2 1 )</td>
<td>( \beta_2 1 )</td>
<td>...</td>
</tr>
<tr>
<td>( \beta_4 )</td>
<td>( \beta_3 0 )</td>
<td>( \beta_3 0 )</td>
<td>( \beta_3 0 )</td>
<td>( \beta_3 0 )</td>
<td>...</td>
</tr>
<tr>
<td>( \beta_5 )</td>
<td>( \beta_4 0 )</td>
<td>( \beta_4 0 )</td>
<td>( \beta_4 0 )</td>
<td>( \beta_4 0 )</td>
<td>...</td>
</tr>
<tr>
<td>( \beta_6 )</td>
<td>( \beta_5 0 )</td>
<td>( \beta_5 0 )</td>
<td>( \beta_5 0 )</td>
<td>( \beta_5 0 )</td>
<td>...</td>
</tr>
<tr>
<td>( \beta_7 )</td>
<td>( \beta_6 0 )</td>
<td>( \beta_6 0 )</td>
<td>( \beta_6 0 )</td>
<td>( \beta_6 0 )</td>
<td>...</td>
</tr>
<tr>
<td>( \beta_8 )</td>
<td>( \beta_7 0 )</td>
<td>( \beta_7 0 )</td>
<td>( \beta_7 0 )</td>
<td>( \beta_7 0 )</td>
<td>...</td>
</tr>
</tbody>
</table>

**Motivation**

**Overview**

**Contributions**

**Approach in Detail**

**Evaluation**

---

*every n seconds, select a composition that we either (1) know little about; or (2) know performs well*
### Approach in Detail — Learning Module

- **An online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*.

<table>
<thead>
<tr>
<th>β₀</th>
<th>TPC.Plain</th>
<th>PT.Plain</th>
<th>TPC.Com.GZip</th>
<th>TPC.Com.ZLib</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>β₁&lt;sub&gt;ReqPT&lt;/sub&gt;</td>
<td>β₀ 1</td>
<td>β₀ 1</td>
<td>β₀ 1</td>
<td>β₀ 1</td>
<td>...</td>
</tr>
<tr>
<td>β₂&lt;sub&gt;HTTPCom&lt;/sub&gt;</td>
<td>β₀ 1</td>
<td>β₁ 1</td>
<td>β₁ 0</td>
<td>β₁ 0</td>
<td>...</td>
</tr>
<tr>
<td>β₃&lt;sub&gt;HTTPCoCache&lt;/sub&gt;</td>
<td>β₀ 0</td>
<td>β₂ 0</td>
<td>β₁ 0</td>
<td>β₂ 0</td>
<td>...</td>
</tr>
<tr>
<td>β₄&lt;sub&gt;HTTPCache&lt;/sub&gt;</td>
<td>β₀ 0</td>
<td>β₃ 0</td>
<td>β₃ 0</td>
<td>β₃ 0</td>
<td>...</td>
</tr>
<tr>
<td>β₅&lt;sub&gt;ComZLib&lt;/sub&gt;</td>
<td>β₄ 0</td>
<td>β₄ 0</td>
<td>β₄ 0</td>
<td>β₄ 0</td>
<td>...</td>
</tr>
<tr>
<td>β₆&lt;sub&gt;CacheLRU&lt;/sub&gt;</td>
<td>β₅ 0</td>
<td>β₅ 0</td>
<td>β₅ 0</td>
<td>β₅ 0</td>
<td>...</td>
</tr>
<tr>
<td>β₇&lt;sub&gt;CacheMRU&lt;/sub&gt;</td>
<td>β₆ 0</td>
<td>β₆ 0</td>
<td>β₆ 0</td>
<td>β₆ 0</td>
<td>...</td>
</tr>
<tr>
<td>β₈&lt;sub&gt;CacheLRU&lt;/sub&gt;</td>
<td>β₇ 0</td>
<td>β₇ 0</td>
<td>β₇ 0</td>
<td>β₇ 0</td>
<td>...</td>
</tr>
</tbody>
</table>

![Graph](selectable_window.png)
**Approach in Detail — Learning Module**

- **An online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*.

```
|   | TPC.Plain | PT.Plain | TPC.Com.GZip | TPC.Com.ZLib | ...
|---|-----------|----------|--------------|--------------|---
| $\beta_0$ | $\beta_0$ | $\beta_0$ | $\beta_0$ | $\beta_0$ | ...
| $\beta_1$ | $\beta_1$ | $\beta_1$ | $\beta_1$ | $\beta_1$ | ...
| $\beta_2$ | $\beta_2$ | $\beta_2$ | $\beta_2$ | $\beta_2$ | ...
| $\beta_3$ | $\beta_3$ | $\beta_3$ | $\beta_3$ | $\beta_3$ | ...
| $\beta_4$ | $\beta_4$ | $\beta_4$ | $\beta_4$ | $\beta_4$ | ...
| $\beta_5$ | $\beta_5$ | $\beta_5$ | $\beta_5$ | $\beta_5$ | ...
| $\beta_6$ | $\beta_6$ | $\beta_6$ | $\beta_6$ | $\beta_6$ | ...
| $\beta_7$ | $\beta_7$ | $\beta_7$ | $\beta_7$ | $\beta_7$ | ...
| $\beta_8$ | $\beta_8$ | $\beta_8$ | $\beta_8$ | $\beta_8$ | ...
```

- **Motivation**
- **Overview**
- **Contributions**
- **Approach in Detail**
- **Evaluation**

select: observe reward, update estimates
**APPROACH IN DETAIL — LEARNING MODULE**

- An online learning algorithm which helps solve the *search space explosion*, and balances *exploration* and *exploitation*.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>β₀</th>
<th>β₁</th>
<th>β₂</th>
<th>β₃</th>
<th>β₄</th>
<th>β₅</th>
<th>β₆</th>
<th>β₇</th>
<th>β₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC.Plain</td>
<td>β₀</td>
<td>β₀</td>
<td>β₀</td>
<td>β₀</td>
<td>β₀</td>
<td>β₀</td>
<td>β₀</td>
<td>β₀</td>
<td>β₀</td>
</tr>
<tr>
<td>PT.Plain</td>
<td>β₁</td>
<td>β₁</td>
<td>β₁</td>
<td>β₁</td>
<td>β₁</td>
<td>β₁</td>
<td>β₁</td>
<td>β₁</td>
<td>β₁</td>
</tr>
<tr>
<td>TPC.Com.GZip</td>
<td>β₂</td>
<td>β₂</td>
<td>β₂</td>
<td>β₂</td>
<td>β₂</td>
<td>β₂</td>
<td>β₂</td>
<td>β₂</td>
<td>β₂</td>
</tr>
<tr>
<td>TPC.Com.ZLib</td>
<td>β₃</td>
<td>β₃</td>
<td>β₃</td>
<td>β₃</td>
<td>β₃</td>
<td>β₃</td>
<td>β₃</td>
<td>β₃</td>
<td>β₃</td>
</tr>
</tbody>
</table>

Motivation

Overview

Contributions

Approach in Detail

Evaluation

| select | share estimates across configurations | selectable window |
**APPROACH IN DETAIL — LEARNING MODULE**

- **An online learning algorithm** which helps solve the *search space explosion*, and balances *exploration* and *exploitation*.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>TPC.Plain</th>
<th>PT.Plain</th>
<th>TPC.Com.GZip</th>
<th>TPC.Com.ZLib</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_0$</td>
<td>$\beta_0$</td>
<td>$\beta_0$</td>
<td>$\beta_0$</td>
<td>$\beta_0$</td>
<td>...</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>$\beta_1$</td>
<td>$\beta_1$</td>
<td>$\beta_1$</td>
<td>$\beta_1$</td>
<td>...</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>$\beta_2$</td>
<td>$\beta_2$</td>
<td>$\beta_2$</td>
<td>$\beta_2$</td>
<td>...</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>$\beta_3$</td>
<td>$\beta_3$</td>
<td>$\beta_3$</td>
<td>$\beta_3$</td>
<td>...</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>$\beta_4$</td>
<td>$\beta_4$</td>
<td>$\beta_4$</td>
<td>$\beta_4$</td>
<td>...</td>
</tr>
<tr>
<td>$\beta_5$</td>
<td>$\beta_5$</td>
<td>$\beta_5$</td>
<td>$\beta_5$</td>
<td>$\beta_5$</td>
<td>...</td>
</tr>
<tr>
<td>$\beta_6$</td>
<td>$\beta_6$</td>
<td>$\beta_6$</td>
<td>$\beta_6$</td>
<td>$\beta_6$</td>
<td>...</td>
</tr>
<tr>
<td>$\beta_7$</td>
<td>$\beta_7$</td>
<td>$\beta_7$</td>
<td>$\beta_7$</td>
<td>$\beta_7$</td>
<td>...</td>
</tr>
<tr>
<td>$\beta_8$</td>
<td>$\beta_8$</td>
<td>$\beta_8$</td>
<td>$\beta_8$</td>
<td>$\beta_8$</td>
<td>...</td>
</tr>
</tbody>
</table>

Motivation | Overview | Contributions | Approach in Detail | Evaluation | Barry Porter
---|---|---|---|---|---
OSDI 2016 | | | [34/41] | |
EVALUATION

Adaptation speed

Performance ground truth

Learning characteristics
EVALUATION — LEARNING CHARACTERISTICS

• Using our web server as an example, we evaluate how optimal systems emerge over time in our approach.

• We use a set of different workloads (client request patterns), including synthetic and real-world traces.

• The only data available for learning is (i) the configuration set; and (ii) the metrics and events that the system emits (in this case, response times and request types/volumes).
**Evaluation — Learning Characteristics**

**Key result:** convergence on optimal solution happens much faster than exhaustive search

Distance from optimal solution over time, averaged across 1,000 runs

- **Workload: small text files**
- **Workload: large image files**

Exhaustive approach finds optimal here.
**Evaluation — Learning Characteristics**

**Key result:** once learned, workload changes are rapidly detected and adjusted to

Workload: cycling between two different request patterns every 100 seconds

Distance from optimal solution over time, averaged across 1,000 runs
EVALUATION — LEARNING CHARACTERISTICS

Key result: convergence occurs in a highly varying real-world workload trace

Workload: real-world workload taken from a publicly available NASA web server trace

Distance from optimal solution over time, averaged across 1,000 runs
EVALUATION — OTHER INSIGHTS

• More broadly, *unexpected* optimal designs that emerged due to machine learning were some of the most interesting results.

• This highlighted cases in which our assumptions were wrong about how a given composition would behave, or examples of programmer error / poor design choices.
SUMMARY

• Presented the idea of emergent software systems as a new solution to system complexity and deployment dynamics

• We use a paradigm of continuous self-assembly, finding optimal systems via automated composition from small building blocks

• Future work: studying more applications (other server types, AI, robotics); automated generation of variants; automated environment classification; distributed emergent systems (e.g. entire datacentre software landscapes)

- download our code at http://www.projectdana.com -