



Software Engineering in der industriellen Praxis (SEIP)

Dr. Ralf S. Engelschall

Declarative Languages

Express the target state
and let the machine figure out the steps.

Markup Languages

Write text intermixed with markup information.



```
foo <em>bar <strong>baz
</strong></em> quux
```

Examples:
Wiki, Markdown, AsciiDoc, SGML, HTML, TeX, R[un]off, reStructuredText, RTF

Configuration Languages

Express complex textual configurations.



```
foo bar quux { baz;
quux id 7; baz }
```

Examples:
INI, XML, SXML, JSON, YAML, TOML, HCL

Rule Languages

Express logic and semantic through complex rules.



```
foo(x, y) <- bar(x, y, z)
AND x < 42 AND z >= 10
```

Examples:
SQL, Datalog/RuleML, OWL/SWRL, RIF

Constraint Languages

Find solutions for complex constraints.



```
foo @ bar(X, Y),
baz(X, Y, _) ==> quux.
```

Examples:
MiniZinc, CHR, OCL, Rego, Z3.

Query Languages

Retrieve information through paths and expressions.



```
// foo / bar [ @baz ==
"xxx" && @quux > 10 ]
```

Examples:
Glob, RegExp, CSS Selector, XPath, YARA, GraphQL, SQL, SPARQL, Cypher, GQL, ASTq

Validation Languages

Parse and validate complex textual information.



```
foo ::= "bar(#" (? :
[0-9a-fA-F]{2})+ ")"
```

Examples:
RegExp, Ducky, BNF, PEG, RELAX NG

solution approach: automatically, non-obvious
execution control: automatically, pre-defined
performance optimization: automatically, pre-defined

Imperative Languages

Express the steps
how the machine has to reach the target state.

Shell Languages

Automate execution of system commands.



```
foo -x 2>&1 | bar -y
--quux <(cat *.cf)
```

Examples:
Korn-Shell, Bourne-Shell, Bash, C-Shell, Batch-Script, PowerShell, AppleScript, DCL

Programming Languages

Execute complex algorithmic steps.



```
for (let i = 0; i < 10;
i++) foo(i, 42)
```

Examples:
JavaScript, TypeScript, Scala, Kotlin, Java, C#, C/C++, Rust, Go, Python, Perl, Ruby, Lua

Text-Processing Languages

Manipulate texts through transformations.



```
/^foo/,/bar.*baz/
s/quux\([0-9]*\) /foo\1/g
```

Examples:
ed, ex, sed, AWK, TXR, XSLT, JSLT

Macro Languages

Pre-process texts with macros.



```
define(`foo', `bar$1baz')
foo(quux)bar
```

Examples:
m4, GPP, CPP, Zoem, ProMac

Expression Languages

Expand path, arithmetic, and boolean expressions.



```
{{ foo.bar[*].baz[42]
.quux + 1 }}
```

Examples:
JQ, YQ, MozJEXL, MathML, JUEL, SpEL

solution approach: manually, obvious
execution control: manually, fine-grained
performance optimization: manually, fine-grained

Template Languages

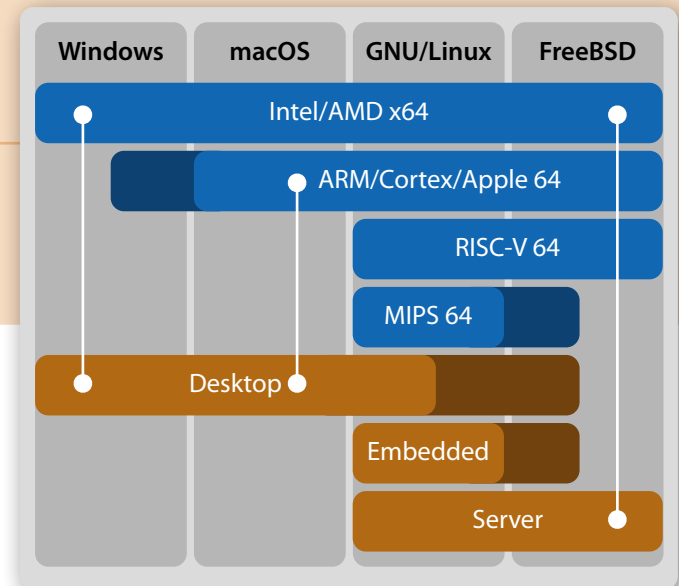
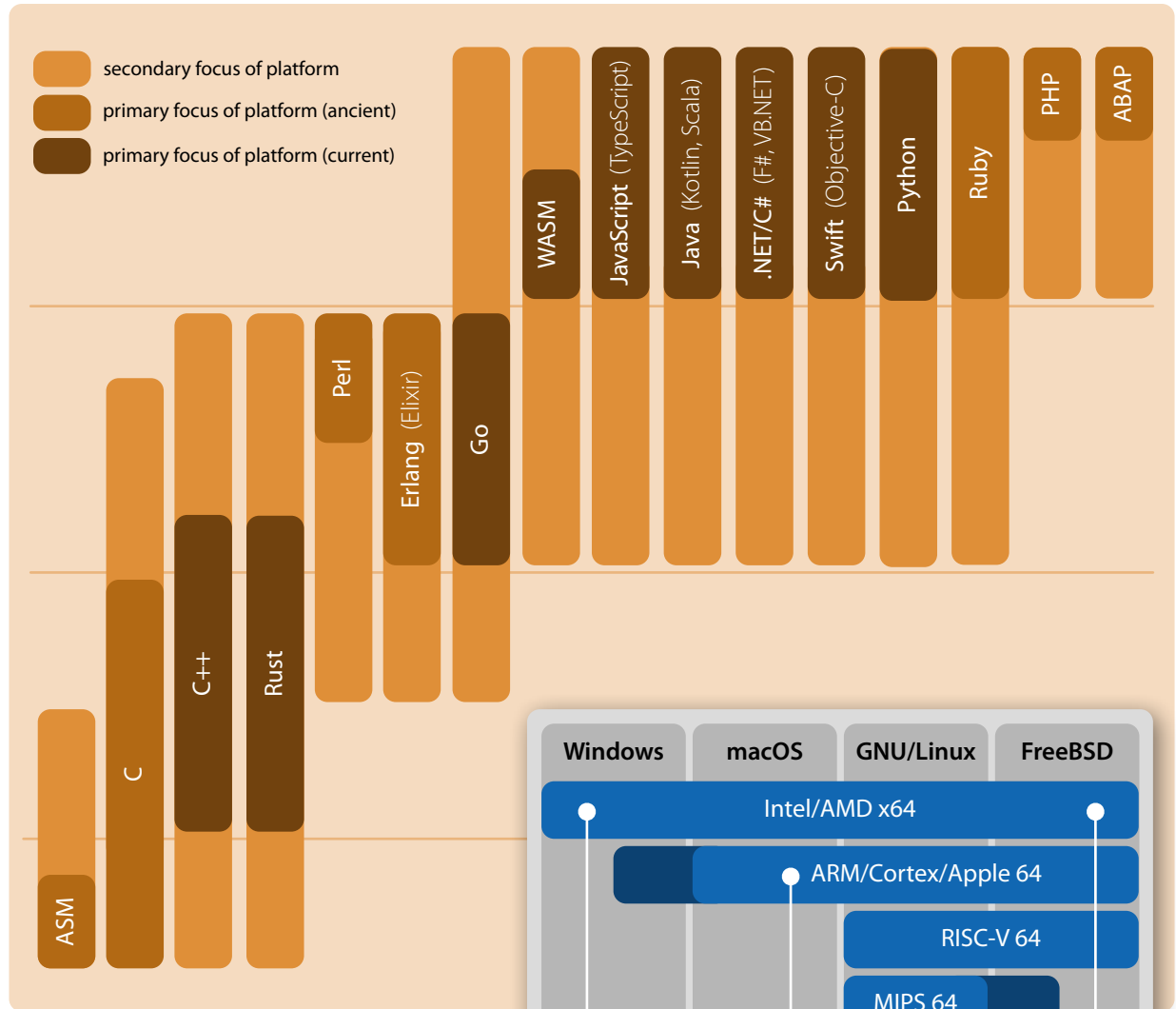
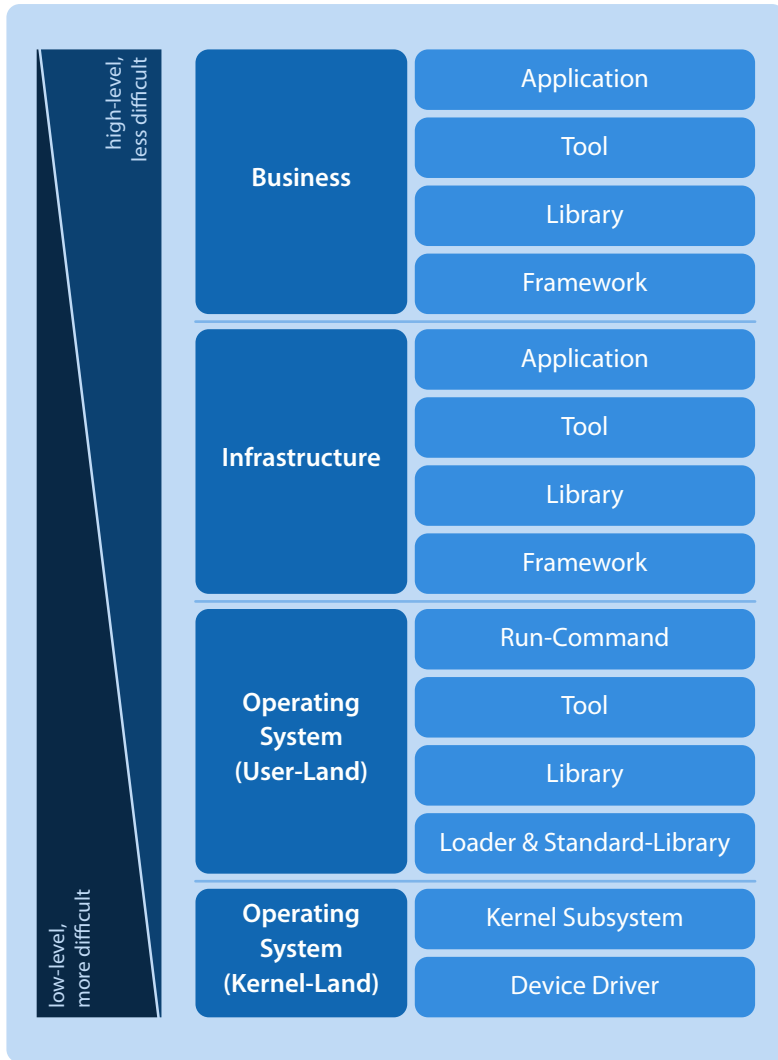
Expand complex text fragments.



```
{% for k, v in items %}
{{k}}: {{v}}{% endfor %}
```

Examples:
Pug, Nunjucks, Handlebars, Mustache, Jinja, Jsonnet

Examples:
essential
recommended
alternative

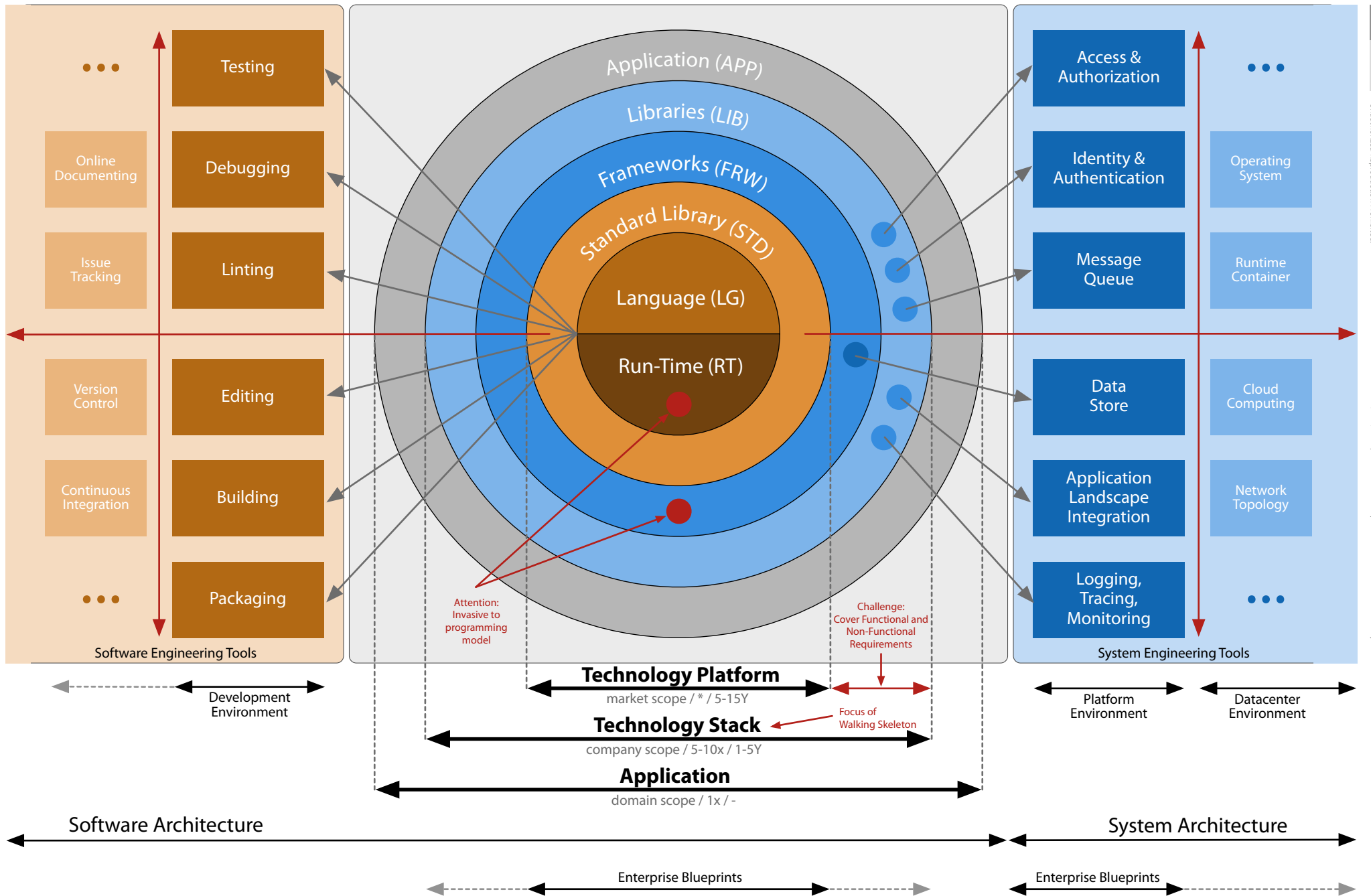


Remember:
A *Technology Platform* is less about choosing a particular programming language and more about choosing a particular ecosystem for targeting a particular level of software!

Opinionated Recommendation (as of 2022):
 Business: **Scala, Kotlin, TypeScript, AssemblyScript**
 Infrastructure: **Go, Rust, Scala, Kotlin, TypeScript**
 Operating System (UL): **Rust, Go**
 Operating System (KL): **C, C++, Rust**


Typical Computing Devices (as of 2022):
 Intel/AMD x64: **Personal Computer (PC)**
 ARM/Cortex/Apple 64: **Raspberry PI, BeagleBone, ROCKSPRO64, iMac**
 RISC-V 64: **Beagle-V, HiFive Unmatched**
 MIPS 64: **Compex WPJ344**

Technology Stack



IT Interface Theme

Style Reset, Shape, Color, Gradient, Shadow, Font, Icon



Bootstrap TypoPRO, FontAwesome, Normalize

IW Interface Widgets


Icon, Label, Text Paragraph, Image, Form, Text-Field, Text-Area, Date Picker, Toggle, Radio Button, Checkbox, Select List, Slider, Progress Bar, Hyperlink, Popup Menu, Dropdown Menu, Toolbar, Tooltip, Tab, Pill, Breadcrumb, Pagination, Badge, Alert, Panel, Modal, Table, Scrollbar, Carousel



Bootstrap Select2, SlickGrid, ...

IL Interface Layouting


Responsive Design, Media Query, Frame, Grid, Padding, Border, Margin, Alignment, Force, Magnetism



Bootstrap Swiper, jQuery Page, ...

IE Interface Effects


Transition, Transformation, Keyframes, Easing Function, Sound Effect, Physics



VueJS Animate.css, DynamicJS, Howler, ...

II Interface Interactions


Mouse, Keyboard, Touchscreen, Gesture, Clipboard, Drag & Drop



VueJS Hammer, Mousetrap, Dragula, ...

IS Interface States


Rendered, Enabled, Visible, Focused, Warning, Error, Floating



VueJS (none)

IM Interface Mask


Markup Loading, Markup Generation, Virtual DOM, Text, Bitmaps, Vectors, 2D/3D Canvas, Accessibility



VueJS jQuery-Markup, D3, Snap.svg, FabricJS, ...

18 Interface Internationalization

Text Internationalization (I18N).



VueJS vue-i18next, I18Nnext

DC Data Conversion

Value Formatting, Value Parsing, Localization (L10N).

\$1,234.56
2016-01-01

VueJS Moment, Numeral, Accounting, ...

DB Data Binding

Reactive, Observer, Unidirectional, Bidirectional, Incremental

<div>FOO</div>
↑ ↓
var bar = "FOO"

VueJS (none)

PM Presentation Model

Parameter Value, Command Value, State Value, Data Value, Event Value, Value Validation, Presentation Logic

```
data: {
  username: String,
  state: {
    username: Enum,
    data: Password,
    state: Password,
    event: LoginRequested: Bool
  }
}
```

ComponentJS (none)

DN Dialog Navigation


Deep Linking, Routing, Dialog Flow

#/foo/123
root → foo → 123

ComponentJS Director, URI.js

DA Dialog Automation

Dialog Macros, Click-Through, Smoke Testing.



ComponentJS ComponentJS-Testdrive

DC Dialog Communication

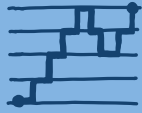
Service, Event, Model, Socket, Hooks



ComponentJS Latching

DL Dialog Life-Cycle

Component States, Component State Transitions.



ComponentJS (none)

DS Dialog Structure

Component, Model/View/Controller Roles, Hierarchical Composition



ComponentJS ComponentJS-MVC

SP State Persistence

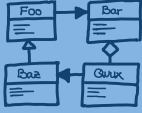
Local Storage, Cookies, Caching



(none) Store.js, JS-Cookie

BM Business Model


Entity, Field, Relationship, Universally Unique Identifiers (UUID)



(none) DataModelJS, Pure-UUID

UA Use-Case Authorization

User Experience, Dialog Restriction, User, Group, Role, Use-Case, Data, Access.



(none) (none)

CN Client Networking


Request/Response, Synchronization, Push, Pull, Pulled-Push, REST, GraphQL, Authentication, Session.

```
query {
  Foo(id: 123) {
    name,
    withBar: {
      id, name,
      hasQuux: {
        id, title } } } } }
```

(none) Axios, Apollo Client

ED Environment Detection

Runtime Detection, Feature Detection.



(none) Modernizr, FeatureJS, jQuery-Stage

ED Environment Detection


Detect the run-time environment, like underlying operating system, execution platform, network topology, feature toggles, etc.



Node | process, syspath

AP Argument Parsing


Parse options and arguments of the Command-Line Interface (CLI) to bootstrap application parameters.



(none) | yargs

CP Configuration Parsing


Load and parse directives from configuration file to bootstrap application parameters.



(none) | js-YAML

PD Process Daemonizing

Detach from the startup terminal and host process in order to run fully independently.



(none) | daemonize2

PM Process Management

(Pre-)fork child processes and/or threads of execution and monitor and control them during the life-cycle of the application.



(none) | cluster, nodemon

CM Component Management

Structure the code into components, instantiate them under run-time and manage them in a stateful component life-cycle.



Microkernel | (none)

CC Component Communication

Provide inter-component communication mechanisms like events, hooks, registry, etc.



Microkernel | Latching

SN Server Networking


Listen to network sockets, accept connections and manage request/response and message communication.



HAPI | hapi-plugin-websocket, ws

PI Peer Information


Determine unique identification and add-on information about the client peer.



HAPI | hapi-plugin-peer, geoip

SH Session Handling

Manage secured per-connection sessions to keep state between communication requests and/or client sessions.



HAPI | YAR

UA User Authentication


Determine and validate the unique identity of the user communicating over the current network connection.



HAPI | JWT, Passport

RV Request Validation


Validate the syntactical and semantical compliance of the requests and sanitize the requests.



HAPI | Joi, DuckyJS

RP Request Processing


Process the request by dispatching execution according to the provided request and determined context information.



HAPI | GraphQL.js

RA Role Authorization


Determine whether the role of the current user is allowed to execute the current request.



(none) | GraphQL-Tools-sequelize

CN Client Networking


Provide mechanisms to connect to peers over the network and perform request/response and/or publish/subscribe communication.



(none) | Axios, MQTT.js, ws

TS Task Scheduling

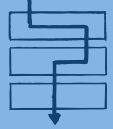
Schedule and execute recurring tasks independent of regular I/O operations.



(none) | node-scheduler

ET Execution Tracing

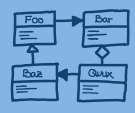
Provide mechanisms for tracing the execution by logging event and measurement information at certain points of interest.



Microkernel | Winston

DA Database Access

Map in-memory domain entities onto data store dependent persistent data structure.



Sequelize | GraphQL-Tools-Sequelize

DC Database Connectivity


Locally or remotely connect the database access layer to the underlying data store.



Sequelize | sqlite3, pg

DS Database Schema

Create, update or downgrade the data schema inside the underlying data store.



Sequelize | (none)

DB Database Bootstrapping

Create, update or downgrade both mandatory bootstrapping and optional domain-specific data inside the underlying data store.



Sequelize | ini

AMA Bare Amalgamation

Manually deploy all applications into a single, shared, and unmanaged filesystem location. Dependencies are resolved manually. Examples: Windows Fonts, Unix 1990th /usr/local.

Pro: simple deployment
Con: incompatibilities, hard uninstallation

UHP Unmanaged Heap

Manually deploy all applications into multiple, distinct, and unmanaged filesystem locations. Dependencies are resolved manually. Examples: macOS *.app, OpenPKG LSYNC.

Pro: simple deployment, easy uninstallation
Con: no repair mechanism

MHP Managed Heap

Let individual installers deploy applications into multiple, distinct, and managed filesystem locations. Dependencies are manually resolved or bundled. Examples: macOS *.pkg, Windows MSI, InnoSetup.

Pro: easy uninstallation, repairable
Con: requires installer, diversity, no dep.

PKG Managed Package

Let a central package manager deploy all applications into a single, shared, and managed filesystem location. Dependencies are automatically resolved. Examples: APT, RPM, FreeBSD pkg, MacPorts, Gradle, NPM.

Pro: easy uninstall., repairable, dependencies
Con: P.M. pre-installation, P.M. single instance

CON Container Image

Bundle an application with its stripped-down OS dependencies and run-time environment into a container image. Examples: Docker/ContainerD, Kubernetes/CRI-O, Windows Portable Apps.

Pro: independent, simple deployment
Con: fewer variations, no dependencies

STK Package/Container Stack

Establish an application out of multiple Managed Packages. Examples: OpenPKG Stack, Docker Compose, Kubernetes/Kompose, Kubernetes/Helm.

Pro: independent, flexible
Con: overhead

VMI Virtual Machine Image

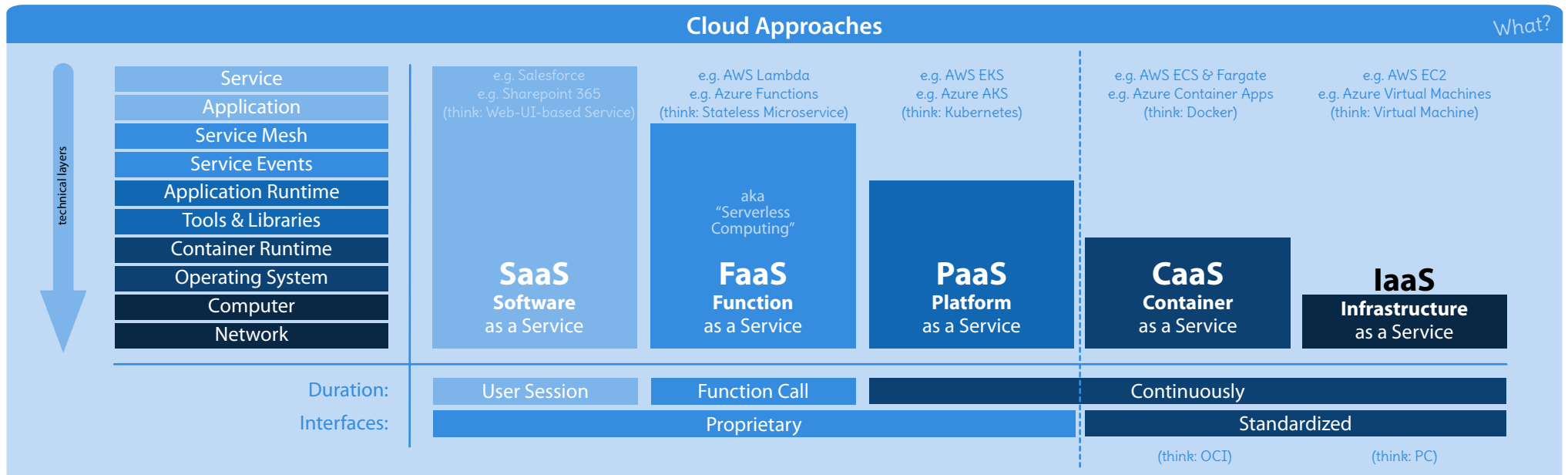
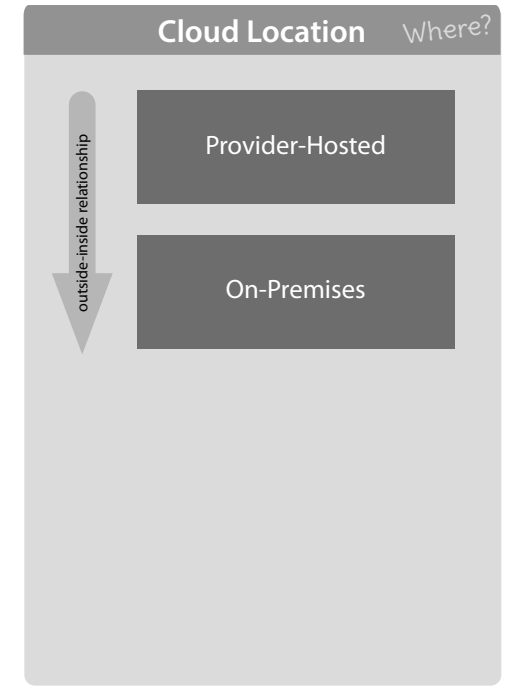
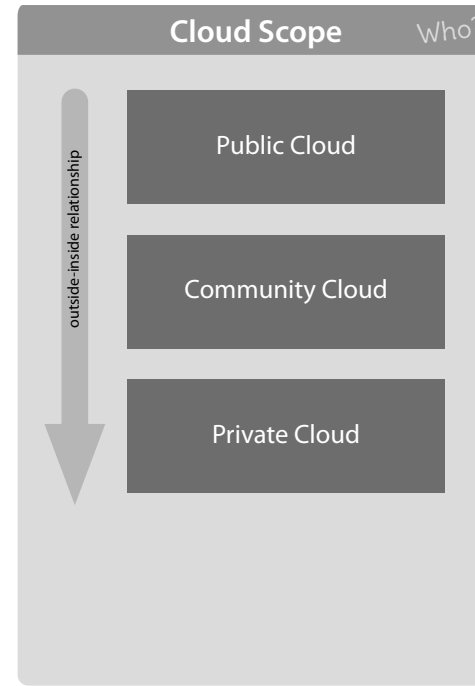
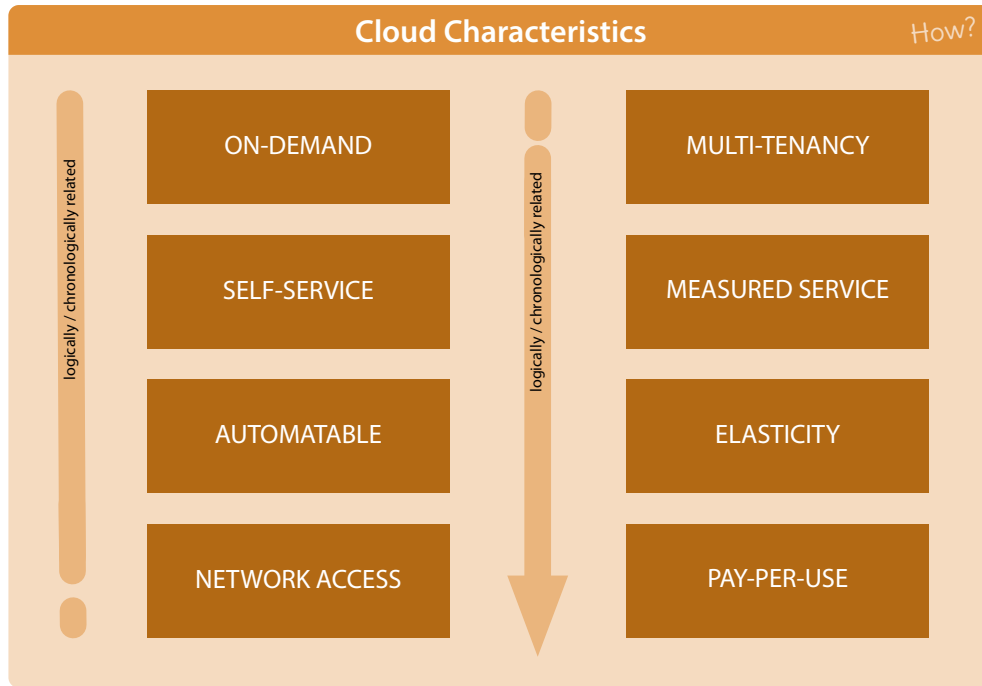
Bundle an application with its full OS dependencies and run-time environment into a virtual machine image and deploy and execute this on a hypervisor. Examples: VirtualBox, VMWare, HyperV, Parallels, QEMU.

Pro: all-in-one, independent
Con: overhead, sealed, inflexible

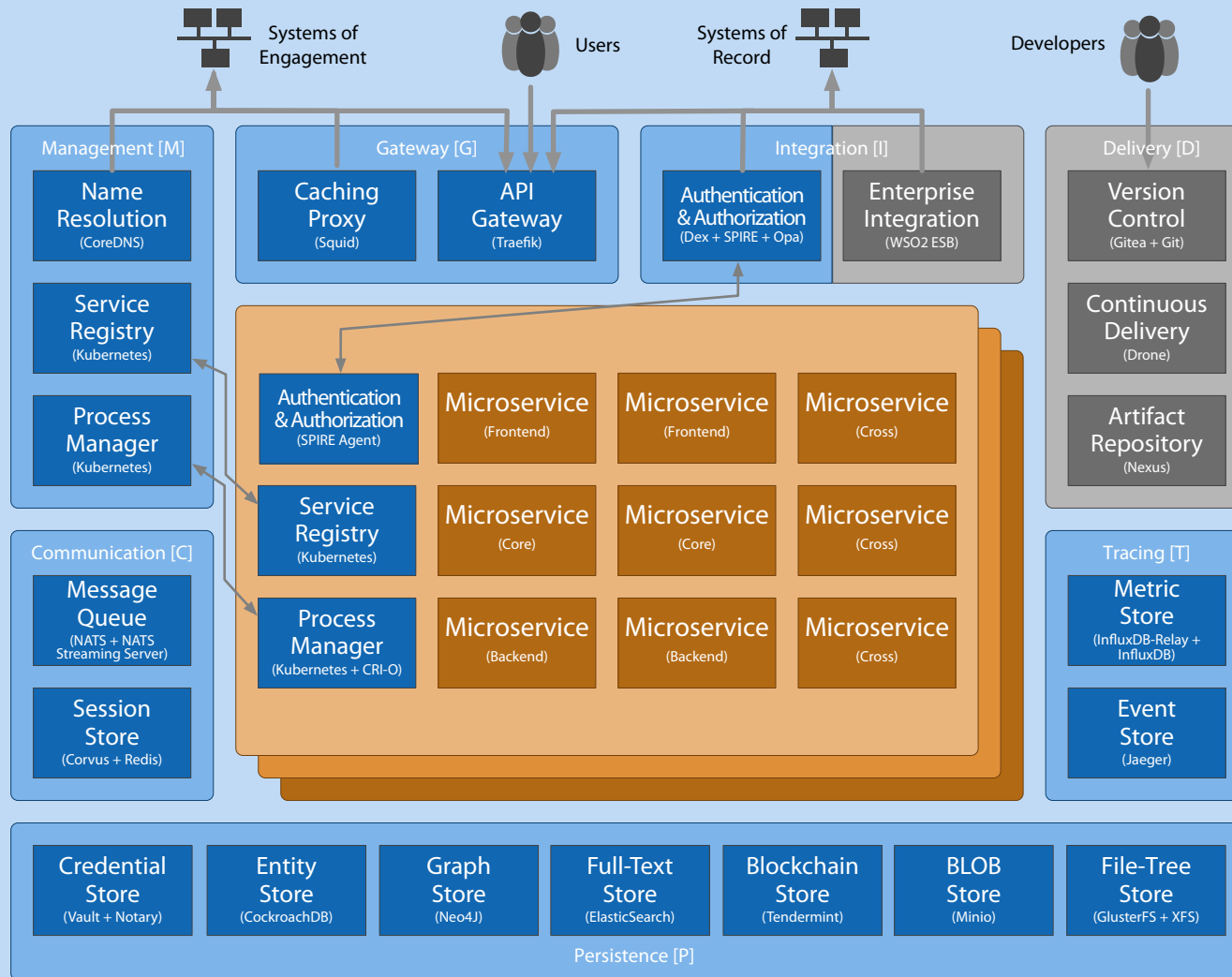
APP Solution Appliance

Bundle an application with its full OS dependencies, run-time environment and underlying hardware. Examples: AVM Fritz! Box, SAP HANA.

Pro: all-in-one, independent
Con: expensive, sealed, inflexible



Reference Architecture Blueprint



- Common Group
- Microservice Group
- (Virtual) Machine
- Platform Application (HA)
- Foreign Application (non-HA)
- Microservice Application (HA)

Major Approach Idea:
With Cloud-Native Architecture one maximizes the leverage of PaaS-like, high-available, and scalable Cloud services at the level of Software- and Systems-Architecture for a whole set of applications.

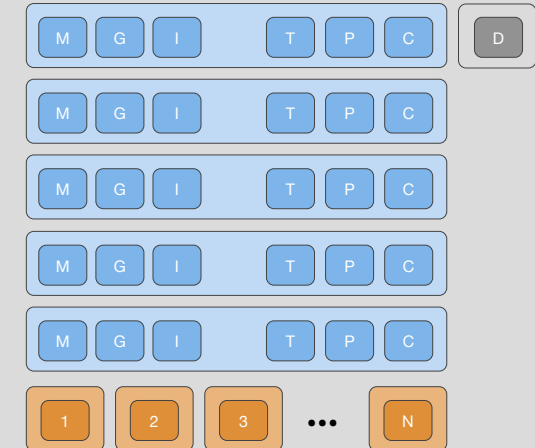
- Major Design Criteria:**
1. Targets DevOps approach.
 2. Targets Continuous Delivery process.
 3. Targets Microservice Architecture.
 4. Targets Container Image deployment.
 5. Targets Service Mesh communication.
 6. Targets Server Cluster setup.
 7. Provides High-Availability of Service Platform
 8. Provides High-Availability of Application Microservices.
 9. Provides Scalability of Application Microservices.

CNCF Cloud-Native Definition 1.0

Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach. These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

Practical Cluster Setups

Standard Cluster Setup (5+1+N Machines):



Partitioned Cluster Setup (5x2+1+N Machines):



GOAL

UX **Exceptional User Experience**


Provide exceptional *User Experience* by taking temporary client/server network offline situations into account and establish trust in the offline capability of the application.



CONTEXT


IS **Business Information Systems**

Client/server applications which drive business processes through use-cases, based primarily on the editing, storing, and retrieving of information.



CC **Cloud Computing**


On-demand availability of computing resources, especially data storage and computing power, without direct active management by the customer.



CHALLENGES


VPN **Toggled Virtual (Private) Networks**

Network offline situations caused by the explicit on/off toggling of overlayed *Virtual (Private) Networks* by the user.




MNC **Switched Mobile Network Cells**

Network offline situations caused by the implicit switching between the mobile network cells by the device during mobile use.



NCO **Failed Network Components**

Network offline situations caused by the failure of any network components between the client and server tiers of an application.




MATURITY LEVELS

L2 **Offline Read**

During offline phase, client allows **read**-operations, but **no write**-operations, to locally cached data.


*Data Access Layer Proxy
Web Storage API*



L5 **Offline Transactional Read / Write**

During offline phase, client allows **non-atomic (transactional)** read/write-operations to any locally cached data.

*SAGA
Compensation*


L1 **Offline Aware****L4** **Offline Read & Atomic Write****L0** **Offline Unaware****L3** **Offline Read & User-Exclusive Write**

Hint: you cannot solve offline scenarios at the technical level if your use-cases are not already aligned to them.

Notice: each level increases user experience, but also increases technical complexity.