4-2020

Rules as code: Seven levels of digitisation

Meng Weng WONG
Singapore Management University, mwwong@smu.edu.sg

RULES *as* CODE

Seven Levels of Digitisation

A guide intended to accelerate sensemaking in discussions involving RaC.

Centre for Computational Law *Singapore Management University*

AUTHORSHIP & COPYRIGHT ATTRIBUTION

This research is supported by National Research Foundation (NRF) Singapore and Infocomm Media Development Authority, Smart Systems Research Programme, under its Industry Alignment Fund – Pre-Positioning Programme, as the Research Programme in Computational Law.

DOCUMENT VERSION

1.0.2	2020-04-05	revised and formatted for print
0.1.0	2020-03-27	first draft

Rules as Code: 7 Levels of Digitisation

A guide intended to accelerate sensemaking in discussions involving RaC.

Why this document exists

Without a common frame of reference, project stakeholders risk talking at cross purposes.

Stakeholders contemplating a "digital transformation" project in the legal domain, such as a "Rules as Code" exercise or a RegTech / SupTech proof-of-concept, may find this document useful to agree on a common vocabulary to facilitate discussion and planning.

To that end, this document classifies "digital transformation" of legal rules into a hierarchy of levels which can be included as terms of reference in planning discussions. While this document is informed by academic discourse, it is intended for practitioners and foregoes the usual citation / footnote style in favour of direct applicability by legal engineers.

In the context of work planning, management can say, "we want to build a Level 3.2 RaC prototype", and the product engineering team would be able to say, "OK, here is roughly the time, resource, and process required for that."

Scope

The legal rules envisaged by this document include relatively black-and-white legislative acts and secondary regulations. They do *not* include "fuzzier" rules originating in the judiciary, which are often phrased in the form of legal principles and doctrines. Think "your dwelling can have 2.5 storeys of no more than 8 meters in height each", not "equity must come with clean hands".

Thanks

Meng would like to take this opportunity to express his gratitude to Alexis N. Chun, his partner, co-founder, and Industry Director at the Centre for Computational Law; Lim How Khang, Centre Director; Goh Yihan, Dean of the School of Law; and Steven Miller, Vice-Provost (Research). Thanks are also due to Pia Andrews for her international leadership of the Rules as Code movement.

Level Zero: Non-Digital

Level 0

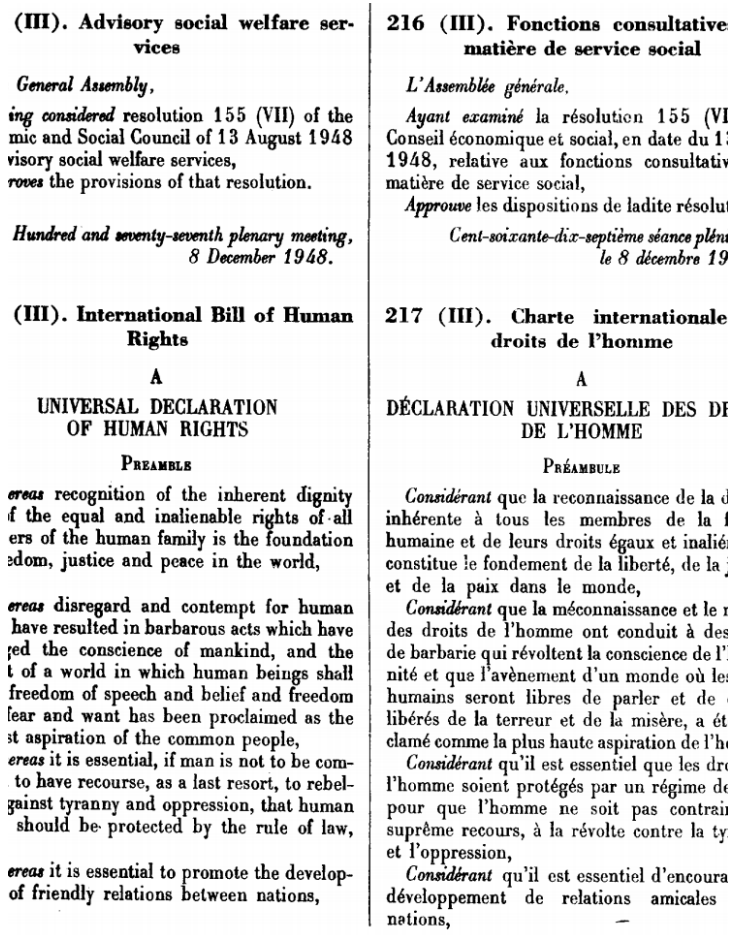
Legislation, regulation, and business rules (LegRegs) are published on paper.

Level One: Digital First Steps

Level 1.0: Scan and OCR

LegRegs have been scanned and digitised by a third party. Some of the content listed at the "Open Access Resources for Law: Primary Sources" page falls into this category. Projects to digitise case law often take this approach.

At this level, a valuable feature is that the text is made searchable and tagged for structure: for example, cross-references are clickable.



(III). Advisory social welfare services

General Assembly,

ing considered resolution 155 (VII) of the mic and Social Council of 13 August 1948 visory social welfare services,
roves the provisions of that resolution.

Hundred and seventy-seventh plenary meeting,
8 December 1948.

(III). International Bill of Human Rights

A

UNIVERSAL DECLARATION OF HUMAN RIGHTS

PREAMBLE

ereas recognition of the inherent dignity f the equal and inalienable rights of all ers of the human family is the foundation edom, justice and peace in the world,

ereas disregard and contempt for human have resulted in barbarous acts which have ed the conscience of mankind, and the t of a world in which human beings shall freedom of speech and belief and freedom fear and want has been proclaimed as the st aspiration of the common people,
ereas it is essential, if man is not to be com- to have recourse, as a last resort, to rebel- gainst tyranny and oppression, that human should be protected by the rule of law,

ereas it is essential to promote the develop- of friendly relations between nations,

216 (III). Fonctions consultative matière de service social

L'Assemblée générale,

Ayant examiné la résolution 155 (VI Conseil économique et social, en date du 1 1948, relative aux fonctions consultativ matière de service social,
Approuve les dispositions de ladite résolu

Cent-soixante-dix-septième séance plén
le 8 décembre 19

217 (III). Charte internationale droits de l'homme

A

DÉCLARATION UNIVERSELLE DES DI DE L'HOMME

PRÉAMBULE

Considérant que la reconnaissance de la d inhérente à tous les membres de la f humaine et de leurs droits égaux et inalié constitue le fondement de la liberté, de la j et de la paix dans le monde,
Considérant que la méconnaissance et le r des droits de l'homme ont conduit à des de barbarie qui révoltent la conscience de l' nité et que l'avènement d'un monde où les humains seront libres de parler et de libérés de la terreur et de la misère, a ét clamé comme la plus haute aspiration de l'h
Considérant qu'il est essentiel que les dro l'homme soient protégés par un régime de pour que l'homme ne soit pas contrai suprême recours, à la révolte contre la ty et l'oppression,
Considérant qu'il est essentiel d'encoura développement de relations amicales nations,

Level 1.1: Clean Digital Publications

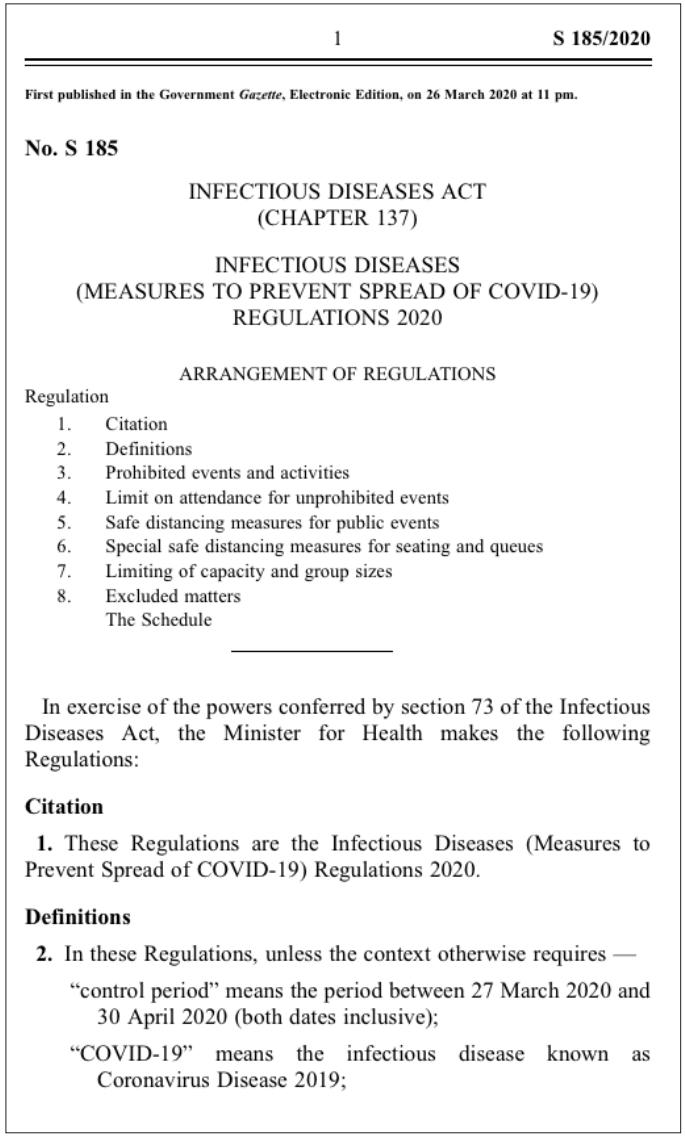
LegRegs are published in HTML, as "born digital" PDFs, or other such formats, available for download or on a website.

Two variants may be distinguished:

- **Level 1.1.1**: access is subject to fee or other restriction, e.g. membership in professional society.
- **Level 1.1.2**: access is open and available without restriction.

These documents are typically authoritative, in the sense that they are published by an authority and may be cited in official filings. Many government legislative assemblies have already achieved this level.

At this level, version history is a valuable feature: what did this legislation look like five years ago?



1 S 185/2020

First published in the Government *Gazette*, Electronic Edition, on 26 March 2020 at 11 pm.

No. S 185

INFECTIOUS DISEASES ACT
(CHAPTER 137)

INFECTIOUS DISEASES
(MEASURES TO PREVENT SPREAD OF COVID-19)
REGULATIONS 2020

ARRANGEMENT OF REGULATIONS

Regulation

1. Citation
2. Definitions
3. Prohibited events and activities
4. Limit on attendance for unprohibited events
5. Safe distancing measures for public events
6. Special safe distancing measures for seating and queues
7. Limiting of capacity and group sizes
8. Excluded matters

The Schedule

In exercise of the powers conferred by section 73 of the Infectious Diseases Act, the Minister for Health makes the following Regulations:

Citation

1. These Regulations are the Infectious Diseases (Measures to Prevent Spread of COVID-19) Regulations 2020.

Definitions

2. In these Regulations, unless the context otherwise requires —

"control period" means the period between 27 March 2020 and 30 April 2020 (both dates inclusive);

"COVID-19" means the infectious disease known as Coronavirus Disease 2019;

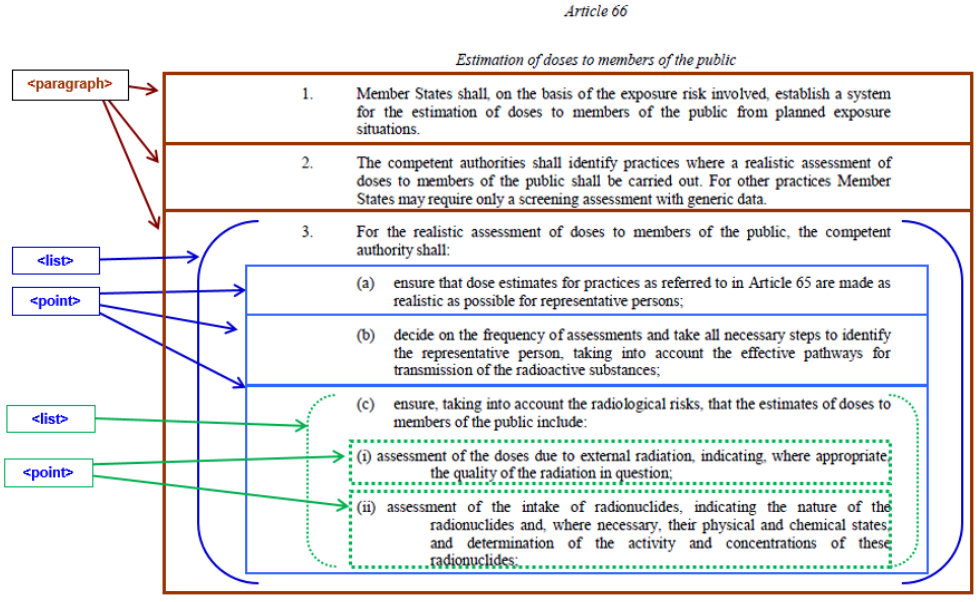
Level 1.2: Formatted for Legal

LegRegs are published in a digital format specialised for legal documents.

Two variants, again:

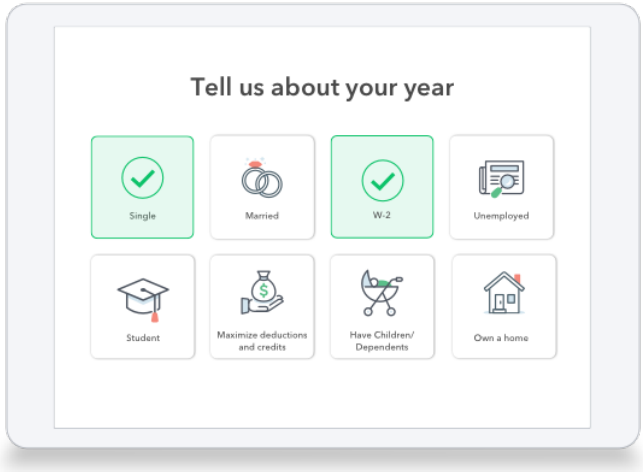
- **Level 1.2.1**: The digital format specification is part of a proprietary toolkit, either developed in-house or available commercially. For example see LegisPro by Xcential.
- **Level 1.2.2**: The digital format specification is an open standard. For example see Akoma Ntoso and LegalDocumentML.

At this level, the structure of the document is explicitly encoded in the format, allowing for precise references. These formats act as "upstream" sources, from which "downstream" HTML and PDF outputs are automatically extracted. The idea of working farther and farther upstream, and letting the computer automatically produce outputs downstream, is a recurring theme in this document. In software engineering, we call this Single Source of Truth.



Level Two: Digital Applications and Products

A private- or public-sector entity has developed application software which embodies a particular set of LegReg rules. For example, in the U.S, Intuit's TurboTax software embodies the tax code, and provides a user-friendly interface to help individuals file annual returns.



In Singapore, IRAS has done much the same, in the back-end of the myTax Portal.

At Level Two, an important feature is that the LegRegs have been faithfully interpreted and usefully "reduced to practice", in the form of an application that helps the user comply with the rules, or otherwise engage with the rules. The application may be delivered over the web or as an installable package or as a mobile app. The application may also take the form of a chatbot that interacts with the user. We call this "accelerating digital service delivery."

Innovative examples include the disruptive products offered by DoNotPay: traditional law firms might not even think of these products as being legal services, but consumers do.

If multiple private-sector vendors develop competing applications, those applications may have differing interpretations. Better for a public sector entity to publish an authoritative engine.

At this level, discussions about digital vs electronic vs paper signatures often arise.

Level 2.1: Spaghetti Rules

At this level, the rules are typically directly encoded in the application, in the form of if-then-else statements written directly in the implementation language, whether front-end or back-end. Some amount of "statutory interpretation" may occur as part of the product development process. The engineering team needs to rework the codebase if the rules change, or if the interpretations change.

This is characteristic of the very first iteration of an RaC system – having a running MVP excuses a multitude of sins!

Level 2.2: Embedded DSL

The rules are modularised into a separate component, tantamount to an embedded DSL, written in the syntax of a host language, typically a general-purpose implementation language. The syntax or file format has developed organically over time.

The rule system may be third-party or home-grown, but there has been an attempt to decouple the rule component from the rest of the software system.

```
class rates_rebates__maximum_income_for_full_rebate(Variable):
    value_type = float
    entity = Titled_Property
    definition_period = YEAR
    label = "Maximum income eligible for the full rebate, less than this number should get full rebate"
    reference = "http://www.legislation.govt.nz/act/public/1973/0005/67.0/DLM409673.html"
    def formula(titled_properties, period, parameters):
        income_threshold = parameters(period).entitlements.rates_rebates.income_threshold
        additional_per_dependant = parameters(period).entitlements.rates_rebates.additional_per_dependant
        initial_contribution = parameters(period).entitlements.rates_rebates.initial_contribution
        # sum allowable income including all the dependants for property
        allowable_income = (titled_properties.sum(titled_properties.members('rates_rebates__dependants', period))
                            * additional_per_dependant) + income_threshold
        # what we're using to compute the maximum salary for full rebate
        rebate = parameters(period).entitlements.rates_rebates.maximum_allowable
        rates_total = titled_properties('rates_rebates__rates_total', period)
        return (((((rates_total - initial_contribution) - rebate)
                - ((rates_total - initial_contribution) / 3)) * 8) + allowable_income)
```

The logic of the original legislation has been translated directly to executable code. Only someone who is a Python programmer and who also has read the legislation will be able to validate this code.

Popular systems include DocAssemble and OpenFisca. Both rely heavily on Python.

The following stanza digitises part of the New Zealand Rates Rebates Act in OpenFisca's Python syntax. Many level 2 implementations look like this:

Level Three:
Declarative Rules with Separate Rule Engines

The rules are implemented in a declarative layer separate from the implementation. A rule engine is explicitly involved in operationalising the rules.

At this level, an important goal is that the rules can be developed and maintained by "non-technical business users" who, supported by the appropriate tooling, can (in theory) formulate rules without needing to work closely with programmers.

In terms of development and drafting process, Levels Two and Three are typically where drafters and policy sources start to work closely with the technologists right from the beginning: see for example the Better Rules Workshop Manual and the Better Rules for Government Discovery Report.

When the rules change, the latest versions can be deployed to the user-facing application interface with a minimum of engineering burden. (At least, that is the vision.)

The Object Management Group is a leading proponent of so-called Model Driven Architecture.

It may not be obvious to an end-user whether an application is built at level 2 or 3, but it is typically very obvious to the product team developing the application.

The essential difference between a Level 2 and a Level 3 system is this:

A Level 2 system has a two-layer architecture:

1. the rules are abstractly specified in natural language (such as English or German), and
2. directly implemented in software code (e.g. Python).

A Level 3 system has a three-layer architecture:

1. based on the natural-language "normative statements",
2. rules are extracted into a relatively abstract machine-readable form, which are read by
3. a rule engine which interprets and executes the rules, as part of a larger software package that interfaces with the user.

Adding the middle layer improves modularity at the cost of additional complexity: now the system includes an external DSL. Creative energy goes into improving the user-interface and user-experience of the middle layer. At a certain point, people feel comfortable referring to that layer as a "low-code" or a "no-code" platform.

At level 3, organisations may choose to publish their rules on a platform such as Github.

Level 3.1: Rule Engine

The rules are expressed in a syntax which is defined by a third-party rule engine: for example, the Drools Rule Language, or Oracle Policy Automation (recently renamed Intelligent Advisor). The rule engine is typically an industry standard piece of infrastructure. Some languages may be implemented in multiple general-purpose languages.

After learning this language, a legal engineer may seek other jobs using the same technology stack, and reuse their skills.

At this level, a valuable feature is that more than one application can consume the same rules. For example, Drools offers a rule API service which could support both a web and a mobile app.

The following example is YAML OpenFisca syntax:

```yaml
- name: "Flax tax on salary - No income"
  period: 2017-01
  input:
    salary: 0
  output:
    flat_tax_on_salary: 0

- name: "Flax tax on salary - With income"
  period: 2017-01
  input:
    salary: 2000
  output:
    flat_tax_on_salary: 500
```

The following example is from Drools:

A Rule in the DROOLS Rule Language

```
package LoanApproval

rule "microfinance"
when
   application: Application(
      loan_amount < 10000,
      duration_year < 5 )
   customer: Customer(
      credit_report == "good" )
then
   application.approval();
   application.setInterest_rate(0.028);
end
```

Level 3.2: Rule Standard

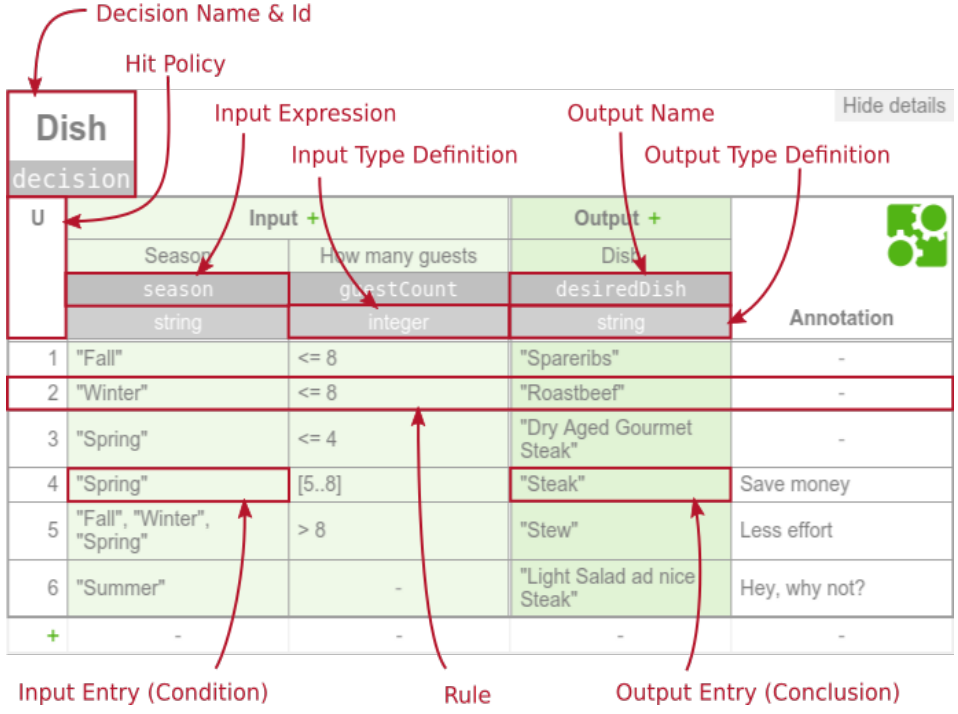
The rules are expressed in a syntax defined by an open standard not tied to a particular implementation, and intended for interoperability between multiple rule engines. Examples include DMN; LegalRuleML; SBVR. In the case of DMN, for example, there are multiple certified, conforming implementations.

After learning this language, a legal engineer may be hired to introduce that technology stack into a new organisation seeking to leapfrog the previous levels.

At this level, a valuable feature is that multiple vendors may bring tooling to the same underlying rule data.

The following example is a decision table expressed in DMN.

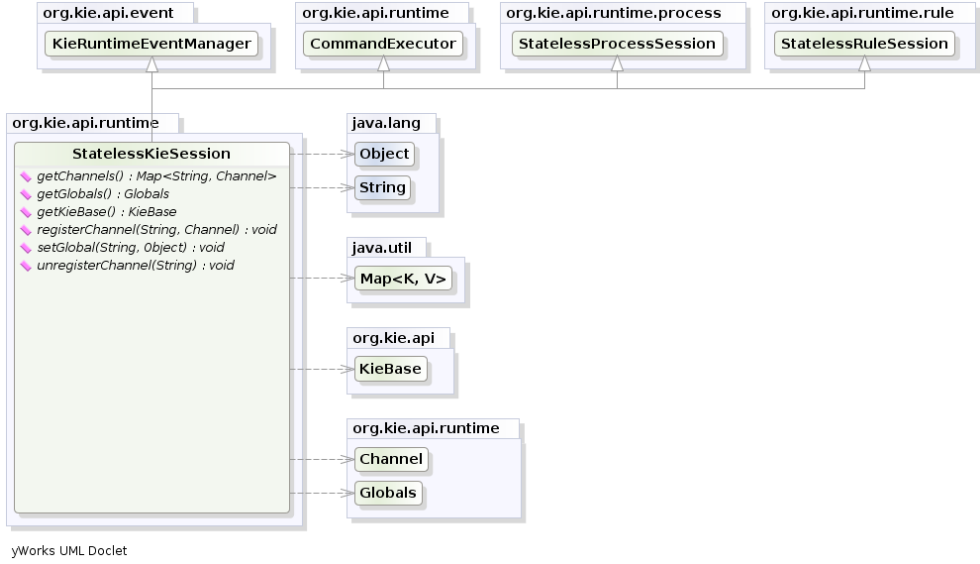
At this level, the rules are typically packaged together with the application, and executed all in one place.



Level 3.3: Rule App

Applications are published as front-end web applications which interface over clearly defined APIs to back-end rule engine services. The APIs are private to the application and not intended for external consumption.

Drools's KIE architecture works this way.

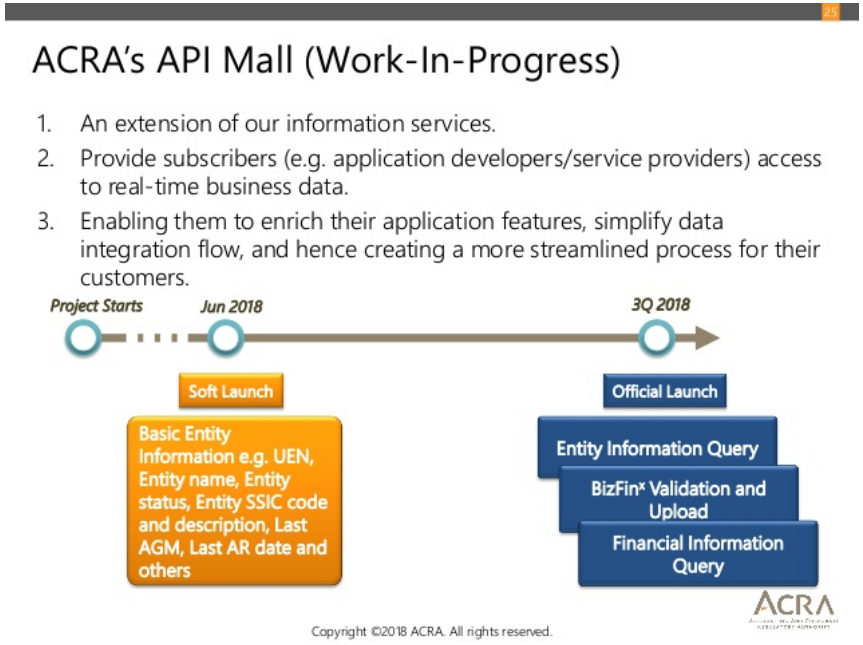


Level 3.4: Rule API

The rule APIs are documented and certified for public consumption; use of the web front end is optional.

Amazon Web Services sets a leading example of API-driven services (in the infrastructure domain, not in legal, but the architecture could work the same way).

Imagine some government department offering an API that says, "submit a proposed set of resolutions; we will return an opinion on whether those resolutions are valid." The API could return warnings such as "er, the directors listed as signatories are not actually the directors we have on record". Or "sorry, the appointment of auditors requires a resolution of the members, not the board." Or "you have passed the deadline for filing this thing, you need to file an additional piece of paper and pay a penalty fee before you can do the actual filing."



Level 3.5: Authoritative API

A rule provider no longer publishes a web front end, but only rule APIs; front-ends become the responsibility of the consumer's preferred interface vendor.

Do responses delivered by the API have the force of law? One could imagine them falling somewhere in between

- an anonymous call to a regulatory body "asking for a friend",
- an official no-action letter, and
- a judgment from a lower court which is open to appeal.

Level 3.6: Rules Only

A rule provider only publishes the rules themselves in a standard format; operating a rule engine against the published rules becomes the responsibility of a third party vendor.

Some rule providers may resist going from level 3.5 to 3.6. They may invent all manner of principled arguments which mask the real reason: they make money off API calls.

Sophisticated rule consumers will demand 3.6, so they can analyse the rules themselves using the tools described under L6.2 below, without exposing the content of their what-if scenarios, and without having to pay per query, or be subject to rate limiting, or have to deal with the risks of an external dependency.

Level Four: About Ontologies

Ontologies are specialised to describe constitutive rules: what entity counts as what role, is a person a natural person or a corporation, is an individual 21 years of age, who is a dependent, what is a business day in which country.

By contrast, regulative rules typically deal with events that happen in time; they impose obligations and penalties on entities: if you make payment after a certain deadline, you will be charged additional fees, unless you filed a late-payment notice by some other deadline, etc.

Some rule systems are essentially ontological and can be expressed entirely in an ontology language. Other rule systems are essentially deontological (must / may / shall not) but rely on ontologies to ground the domain of discourse. (Example: Restaurants who want a liquor license must pay X amount by Y date to have the right to sell wine after Z time. But the application has to be made in the name of the business, not the sole proprietor.)

Level 4.1: Underspecified Ontology

The ontology is implicit. The rules assume that everybody has a shared understanding of common terms. This assumption often leads to trouble. Underspecified ontologies are a common source of open texture.

Level 4.2: Defined Terms

The ontology that underlies the rules is expressed explicitly in documentation. This is equivalent to a comprehensive "Defined Terms" section in a legal document.

An architectural boundary exists between the ontology layer and the rule layer, at least at the conceptual level. Similarly, in legal writing, defined terms are capitalised; uncapitalised use of the terms is carefully avoided.

The constitutive rules about "an X counts as a Y when condition Z holds" are explicitly considered and defined, for objects and for relationships.

This part of the *Bürgerliches Gesetzbuch*, the civil code of Germany, reads like an ontology:

Division 2

Things and animals

Section 90
Concept of the thing

Only corporeal objects are things as defined by law.

Section 90a
Animals

Animals are not things. They are protected by special statutes. They are governed by the provisions that apply to things, with the necessary modifications, except insofar as otherwise provided.

Section 91
Fungible things

Fungible things as defined by law are movable things that in business dealings are customarily specified by number, measure or weight.

Section 92
Consumable things

(1) Consumable things as defined by law are movable things whose intended use consists in consumption or in disposal.

(2) Movable things are also regarded as consumable if they are part of a warehouse store or another aggregate of things whose intended use is the disposal of the individual things.

Section 93
Essential parts of a thing

Parts of a thing that cannot be separated without one or the other being destroyed or undergoing a change of nature (essential parts) cannot be the subject of separate rights.

Level 4.3: Explicit, Embedded

The ontology that underlies the rules is explicitly represented in a machine-consumable syntax, typically in the rule language's type system or class/object model.

Here we see an ontology which, unlike the German Civil Code, considers a non-corporeal relation – an Obligation – to be a Thing.

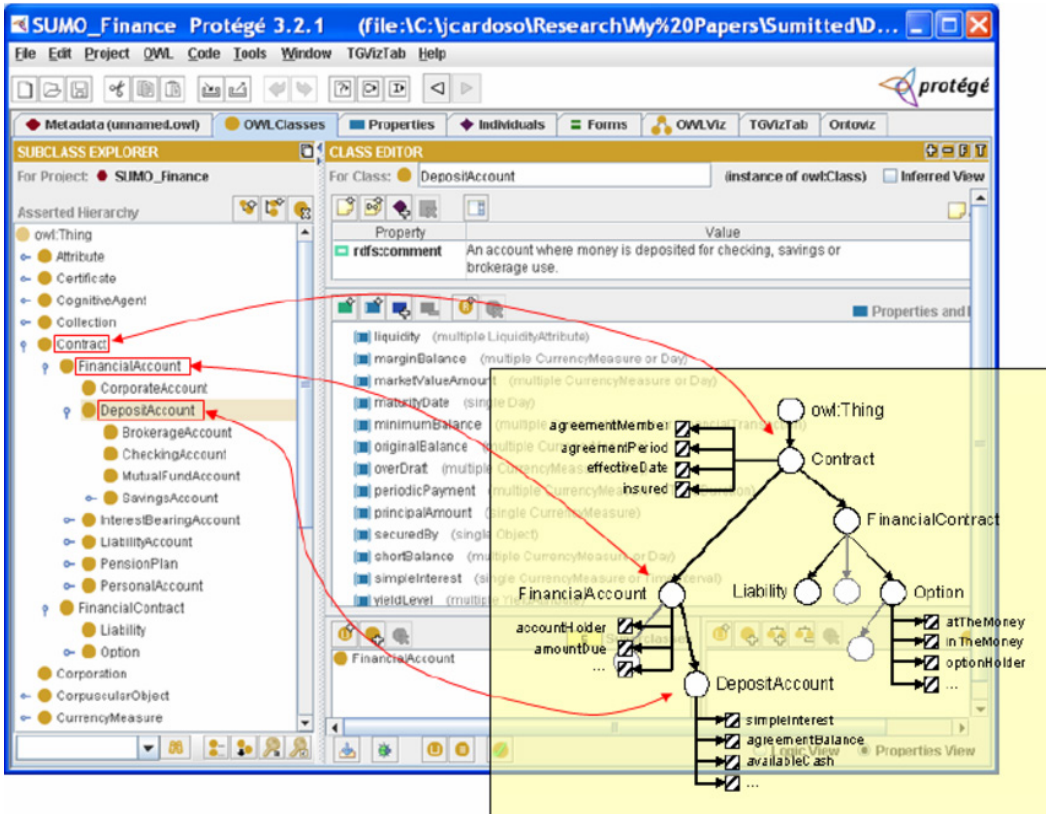
```
Obligation::Thing.
Obligation[|
  obliged_entity=>Thing,
  obliged_action=>Action,
  deadline=>Deadline,
  satisfied=>\boolean
|].
Deadline::Thing.
\date::Deadline.
Relative_Deadline::Deadline[|
  date=>\date,
  duration=>\duration
|].
Employer::Thing.
Determination_of_Year_of_Employment::Thing.
Notification::Thing[|
  sender=>Thing,
  recipient=>Thing,
  content=>Thing,
  date=>\date
|].
Employee::Thing.
Affected_Employee::Thing.
Year_of_Employment::Thing[|
  date_of_commencement=>\date,
  date_of_expiry=>\date
|].
Method_of_Calculation::Thing.
Period_of_Employment::Thing.
```

Credit: Jason Morris, Round Table Law

Level 4.4: Explicit, External

The ontology is expressed using a third-party ontology standard, such as OWL or SUMO. The ontology can be automatically imported for use in the rule language.

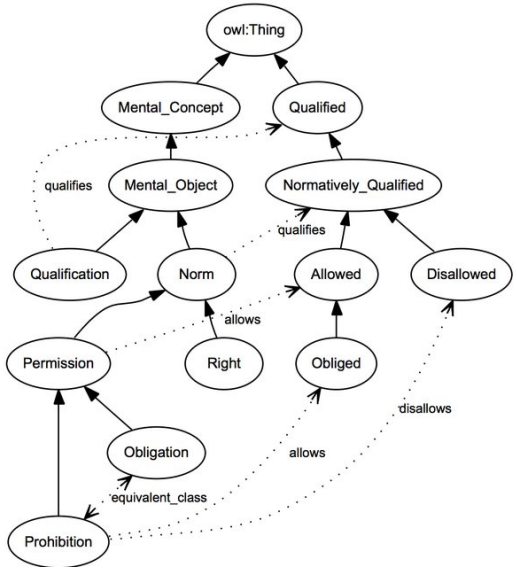
The following screenshot is from the Protégé ontology editor:



Level 4.5: Inheritance

The ontology reuses and inherits an existing standard framework or ontology of legal concepts, such as DOLCE, UFO-L, or LKIF-Core. This opens a whole new can of worms, but maybe the tradeoffs are worth it. For example, a regional economic community (e.g. EU or ASEAN) may opt to harmonise definitions of goods for cross-border trade.

This diagram is from LKIF-Core's ontology of basic legal concepts.



Level Five: Natural Language Generation of "Digital Twins"

Many RaC projects adopt the praxis and ethos of software requirements engineering. As they say, "the devil is in the details." The need to develop detailed machine-consumable rules in parallel with natural language rules forces explicit design discussions earlier than usual. Some refer to this as a "digital twin" strategy. Advanced versions of this strategy could be called "digital first", or "born digital".

The "Better Rules" project conducted in New Zealand, under Pia Andrews's leadership, was the subject of an OECD Observatory of Public Sector Innovation case study. Resources:

- The Practical Better Rules Workshop Manual by Hamish Fraser
- The Better Rules for Government Discovery Report

Computer rules which map exactly to the natural language rules are called "isomorphic".

Level 5.1: Code < Natural

A Natural Language Generation system exists that compiles LegRegs to readable natural language text, with sufficient accuracy that the job of the legislative drafter is significantly transformed from doing the drafting, to reviewing and polishing the machine-produced draft. In the spirit of kaizen, project energy goes toward improving the process toolchain – the NLG engine – rather than the work product itself.

At this level, both drafters and the public consider the natural language version to be the official version.

Level 5.2: Code = Natural

The NLG system compiles LegRegs to readable natural language text, with sufficient fidelity and appropriate tone such that human-readable legislation is automatically compiled from the machine-authored "digital twin" source, which is considered (by the legislative drafters and policy officers, at least) to be the canonical representation. No manual edits to the output are needed.

At this level, the public considers the natural language version to be official, but the drafters privately treat the digital version as canonical, in the same way that programmers treat the source code of a program, not the compiled executable, as canonical.

Level 5.3: Code > Natural

At this level, both drafters and the public consider the digital version to be the canonical, official representation. Welcome to Westworld.

Level Six: Tooling Automation

Level 6.1: Automated KRR

The rules are sufficiently formalised for the following systems to rely on them:

- **Expert systems** support interactive Q&A, with explanation support.
- **Planners** deduce necessary courses of action.
- **Document assembly systems** produce appropriate documents for execution.
- **Contract lifecycle management systems** track dates, obligations, and notices.

These KRR (Knowledge Representation and Reasoning) systems can be said to be automatically extracted from the rules.

Level 6.2: Tools for Rules

Tools assist LegReg "developers" and legal engineers in ways that are similar to how tools assist software developers:

- **Integrated Development Environment** (whether graphical or textual) identifies syntax errors
- **"Proof Assistant"** component of the IDE identifies semantic mistakes, starting with type errors
- **"Unit tests"** allow the developer to monitor the effect of proposed changes on specific reference scenarios. In one banking project, over 3000 such tests were written *by lawyers.* (Source: @yourbrutefruit, 2020-04-05)
- **Property-based testing** helps to automate sanity-checking of the rule system
- **Model checking** helps to identify loopholes and dysfunctional interpretations.
- **Version** and **Dependency Management** help control complexity.
- **Adversarial gaming engines** look for loopholes and clever courses of action.

Many of these tools are yet to be developed, but can be adapted from existing products which have demonstrated their usefulness in the software engineering field.

Level Seven: Universal Adoption

Rules and contracts are universally expected to appear as digital code, in the same way that financial projections are expected to appear as digital spreadsheets, photographs are expected to appear as digital JPEGs, architectural blueprints are expected to appear as digital AutoCAD files, scientific journal manuscripts are expected to be submitted in LaTeX.

While there may be minor turbulence over specific file formats (.xls or .xlsx? .gif or .png?), the idea that legal contracts and legislation/regulations can be passed around in machine-consumable form is totally noncontroversial. *How different things are today,* people will marvel, *from ten years ago, when we were still sending Microsoft Word files around.*

Maybe every contract sent as a PDF contains XMP which represents the semantics of the contract.

A company can be construed as the sum of its contracts.

Auditors assisting with due diligence on an acquisition will say, "please upload your corporate repository to the data-taroom"; they will run software to mechanically analyse the files; and the software will return a checklist already filled in with evaluations and explanations. Due diligence takes hours and days, not weeks and months.

Digital legislation and regulations will serve as the basis for a new generation of applications which help users meet their legal needs without having to pay a law firm by the hour. Michael Genesereth calls this "the cop in the backseat".

Will rules as code replace the opportunity for human oversight and judgment? No; designers of systems are exhorted to leave in entry-points for human discretion. But human input can be requested and provided in a much more lightweight manner. A current opposite extreme is an expert witness or *amicus curiae* called at trial.

COLOPHON

This document was drafted in Google Docs and typeset in Adobe InDesign via DocsFlow by Em Software.
The text face is Neue Haas Unica Pro 10/15. The heading face is Equity by Matthew Butterick.