BP3

There's a faster way to do that®

Revolutionizing Operational Decisions

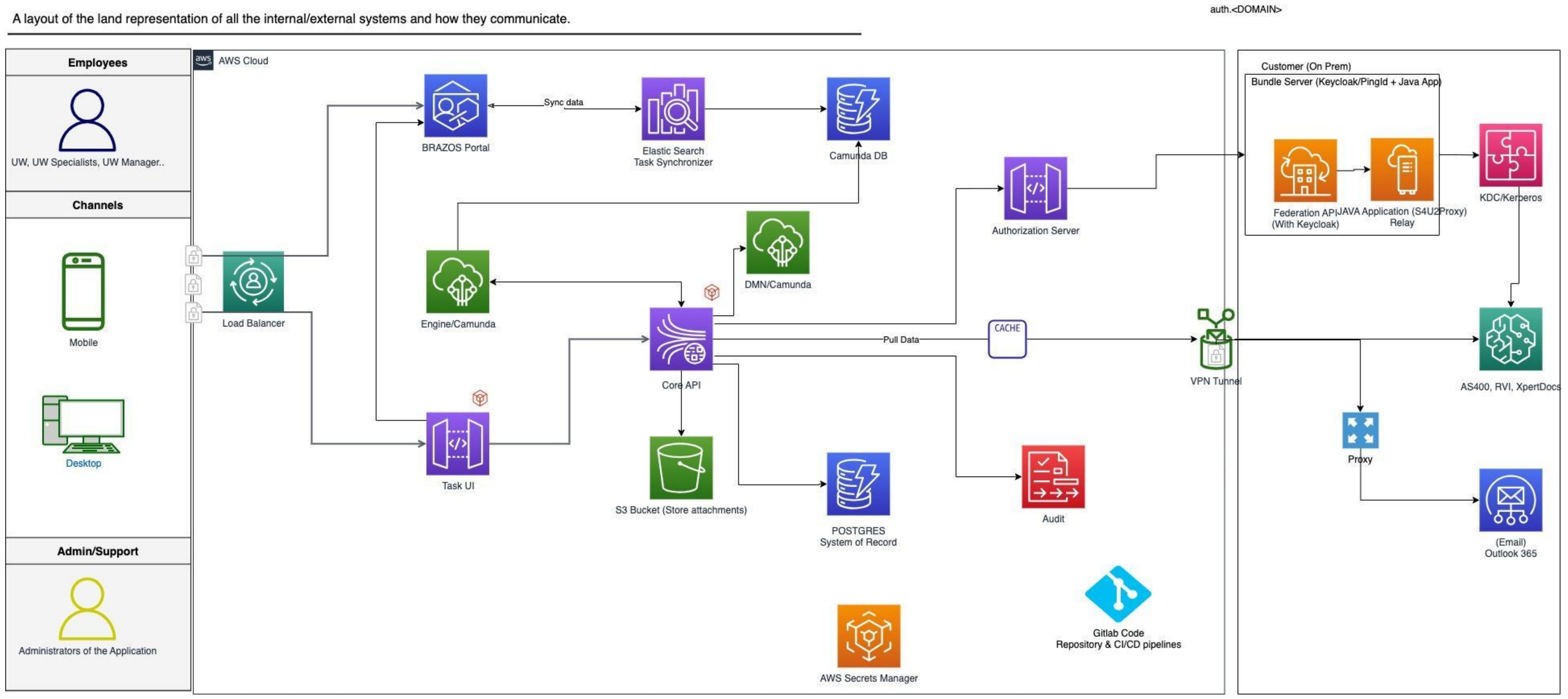
A Medical Liability Insurance Case Study with the Camunda 8 Platform

MEDICAL MALPRACTICE LIABILITY

Application Underwriting Process

APPLICATION MODERNIZATION

From Mainframe to Cloud

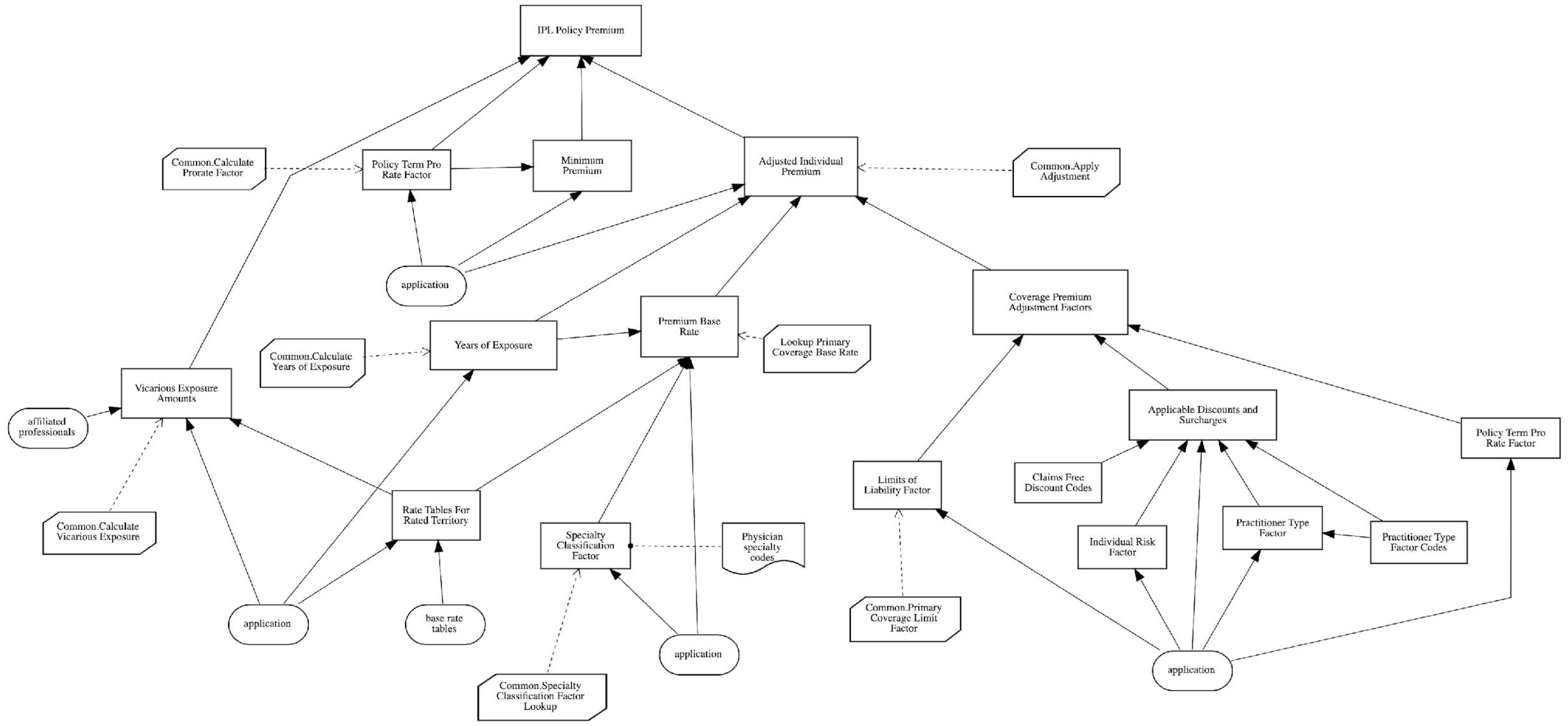


A layout of the land representation of all the internal/external systems and how they communicate.
auth.<DOMAIN>
Employees
UW, UW Specialists, UW Manager..
Channels
Mobile
Desktop
Admin/Support
Administrators of the Application
AWS Cloud
BRAZOS Portal
Sync data
Elastic Search Task Synchronizer
Camunda DB
Load Balancer
Engine/Camunda
DMN/Camunda
Authorization Server
Core API
Pull Data
CACHE
VPN Tunnel
Task UI
S3 Bucket (Store attachments)
POSTGRES System of Record
Audit
Gitlab Code Repository & CI/CD pipelines
AWS Secrets Manager
Customer (On Prem)
Bundle Server (Keycloak/PingId + Java App)
Federation API (With Keycloak)
JAVA Application (S4U2Proxy) Relay
KDC/Kerberos
AS400, RVI, XpertDocs
Proxy
(Email) Outlook 365

There's a faster way to do that.®

LIABILITY POLICY PRICING

Decision Automation with DMN

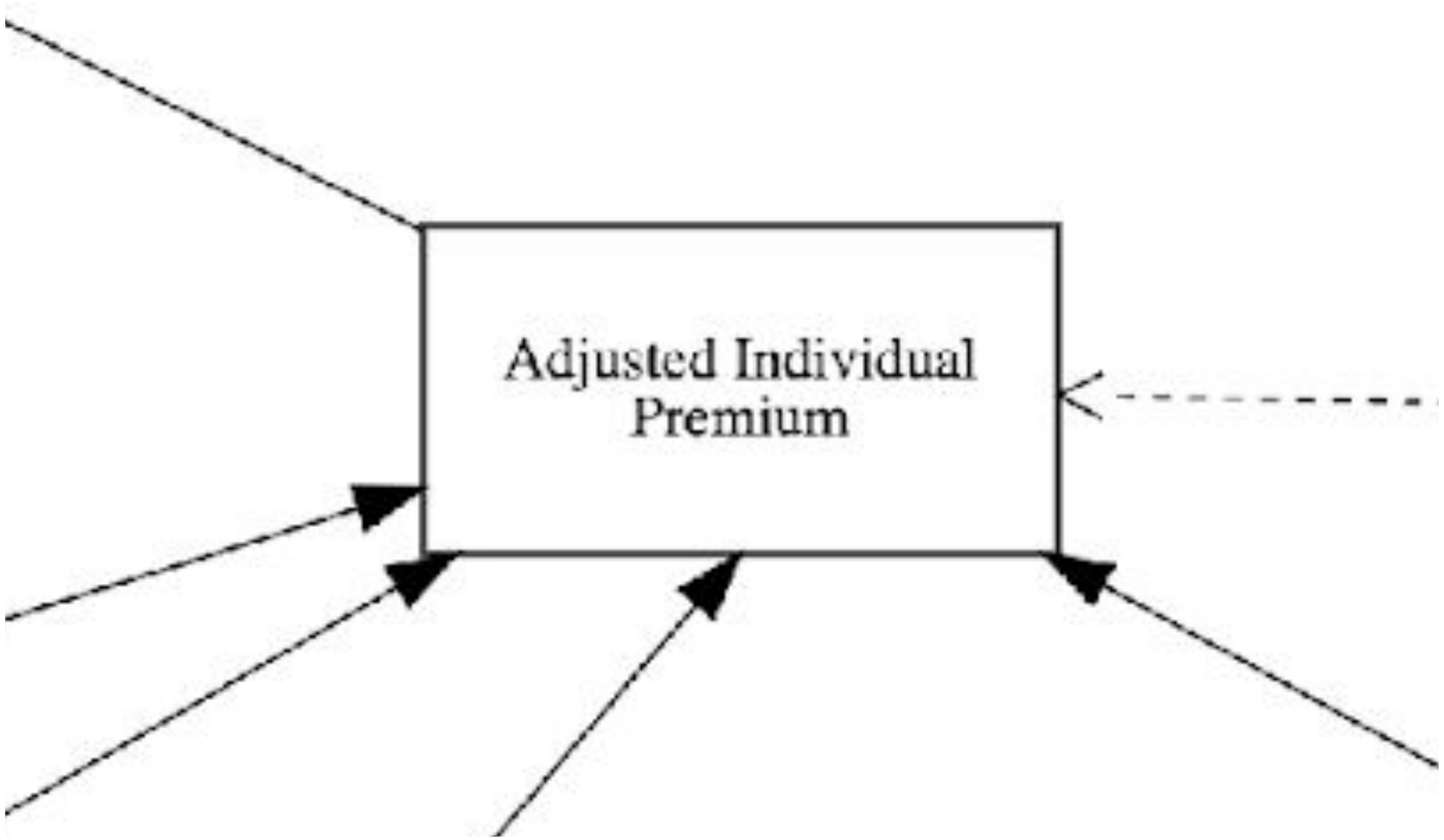


IPL Policy Premium
Common.Calculate Prorate Factor
Policy Term Pro Rate Factor
Minimum Premium
Adjusted Individual Premium
Common.Apply Adjustment
application
Coverage Premium Adjustment Factors
Common.Calculate Years of Exposure
Years of Exposure
Premium Base Rate
Lookup Primary Coverage Base Rate
Vicarious Exposure Amounts
affiliated professionals
Applicable Discounts and Surcharges
Policy Term Pro Rate Factor
Claims Free Discount Codes
Limits of Liability Factor
Rate Tables For Rated Territory
Common.Calculate Vicarious Exposure
Specialty Classification Factor
Physician specialty codes
Practitioner Type Factor
Practitioner Type Factor Codes
Individual Risk Factor
application
base rate tables
Common.Primary Coverage Limit Factor
application
application
Common.Specialty Classification Factor Lookup

There's a faster way to do that.®

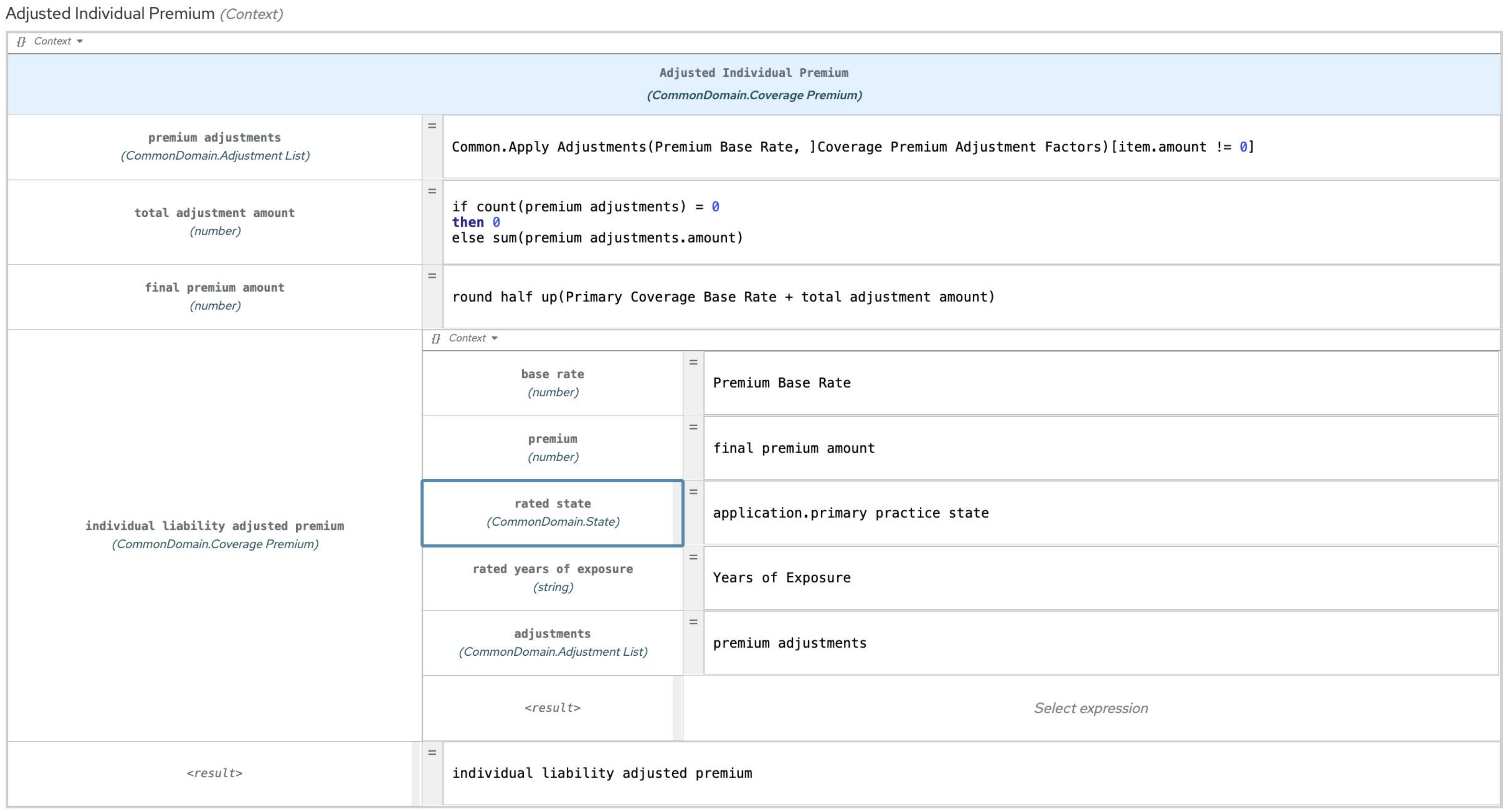
MANAGING COMPLEXITY

From Decision Requirements Diagram to Decisions



MANAGING COMPLEXITY

From Decision Requirements Diagram to Decisions



Adjusted Individual Premium *(Context)*

{} Context

Adjusted Individual Premium *(CommonDomain.Coverage Premium)*		
premium adjustments *(CommonDomain.Adjustment List)*	=	`Common.Apply Adjustments(Premium Base Rate, ]Coverage Premium Adjustment Factors)[item.amount != 0]`

total adjustment amount *(number)*	=	`if count(premium adjustments) = 0`
		`then 0`
		`else sum(premium adjustments.amount)`

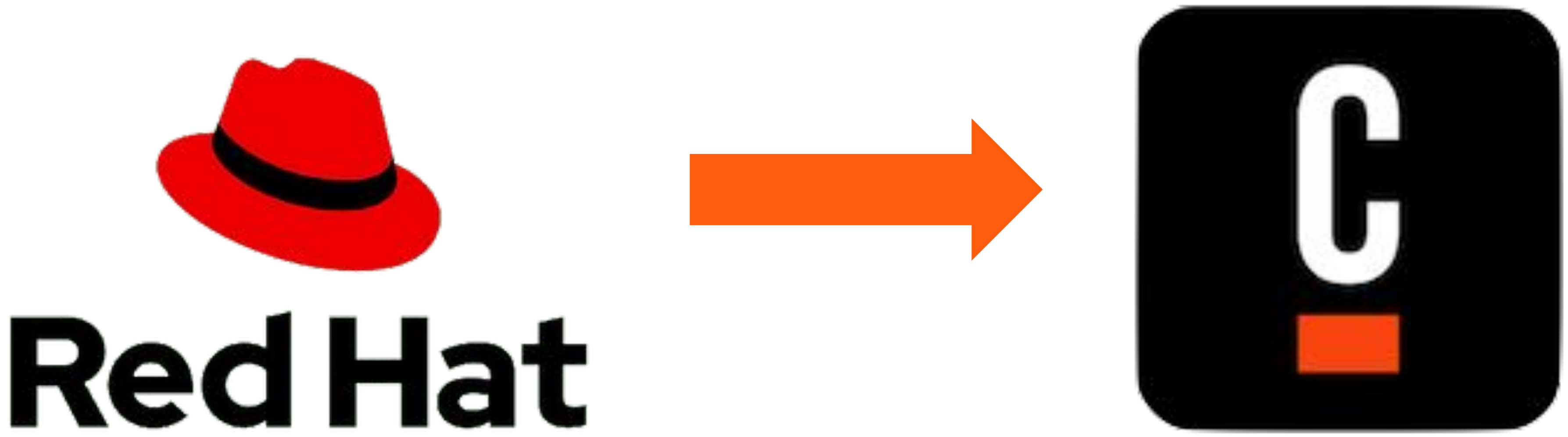
final premium amount *(number)*	=	`round half up(Primary Coverage Base Rate + total adjustment amount)`
individual liability adjusted premium *(CommonDomain.Coverage Premium)*		{} Context
<result>	=	`individual liability adjusted premium`

base rate *(number)*	=	`Premium Base Rate`
premium *(number)*	=	`final premium amount`
rated state *(CommonDomain.State)*	=	`application.primary practice state`
rated years of exposure *(string)*	=	`Years of Exposure`
adjustments *(CommonDomain.Adjustment List)*	=	`premium adjustments`
<result>		*Select expression*

There's a faster way to do that.®

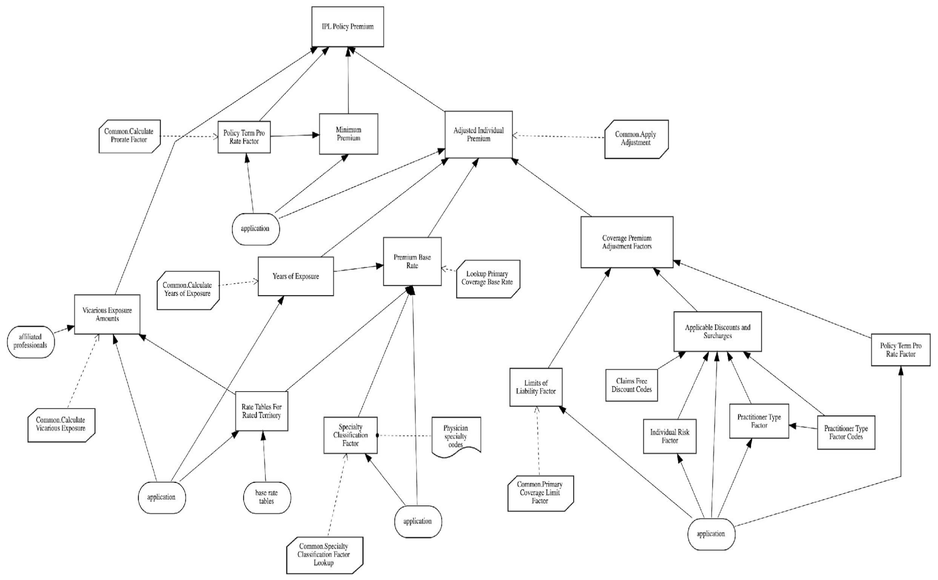
DMN EXECUTION ENGINES

From Red Hat to Camunda 8

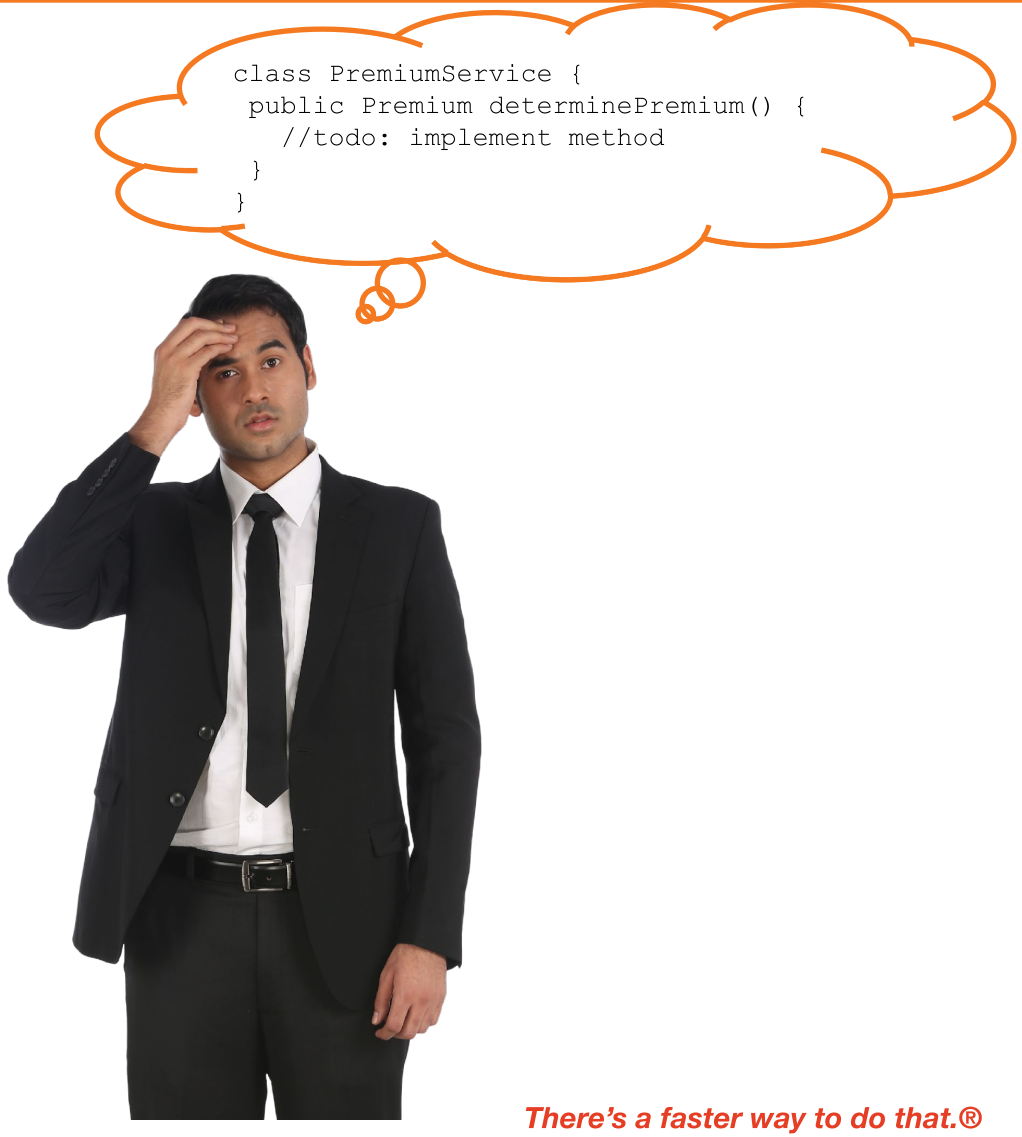


AUTOMATING DECISIONS

From Requirements to Code



IPL Policy Premium
Common.Calculate Prorate Factor
Policy Term Pro Rate Factor
Minimum Premium
Adjusted Individual Premium
Common.Apply Adjustment
application
Coverage Premium Adjustment Factors
Common.Calculate Years of Exposure
Years of Exposure
Premium Base Rate
Lookup Primary Coverage Base Rate
Vicarious Exposure Amounts
affiliated professionals
Applicable Discounts and Surcharges
Policy Term Pro Rate Factor
Limits of Liability Factor
Claims Free Discount Codes
Common.Calculate Vicarious Exposure
Rate Tables For Rated Territory
Specialty Classification Factor
Physician specialty codes
Practitioner Type Factor
Practitioner Type Factor Codes
Individual Risk Factor
application
base rate tables
application
Common.Primary Coverage Limit Factor
application
Common.Specialty Classification Factor Lookup

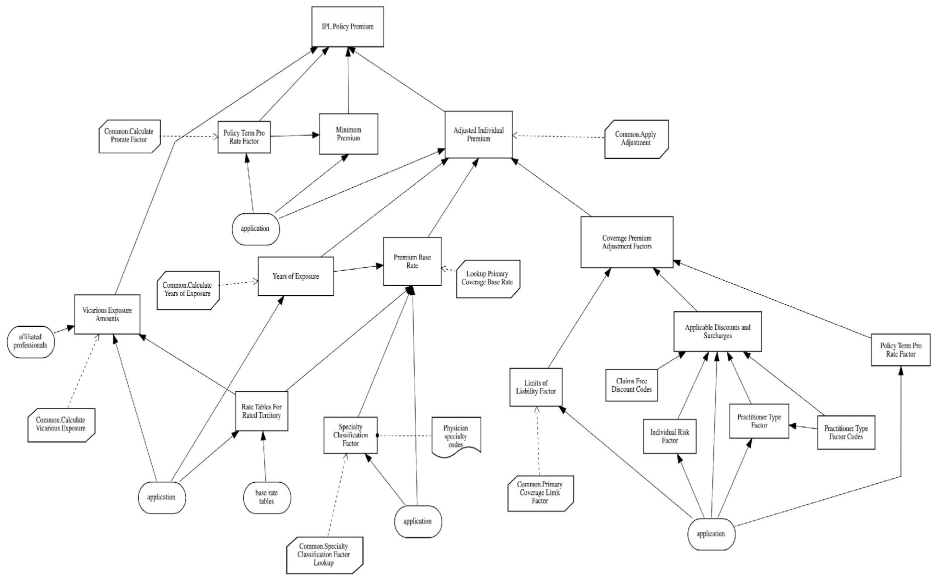


```
class PremiumService {
 public Premium determinePremium() {
   //todo: implement method
  }
}
```

There's a faster way to do that.®

AUTOMATING DECISIONS

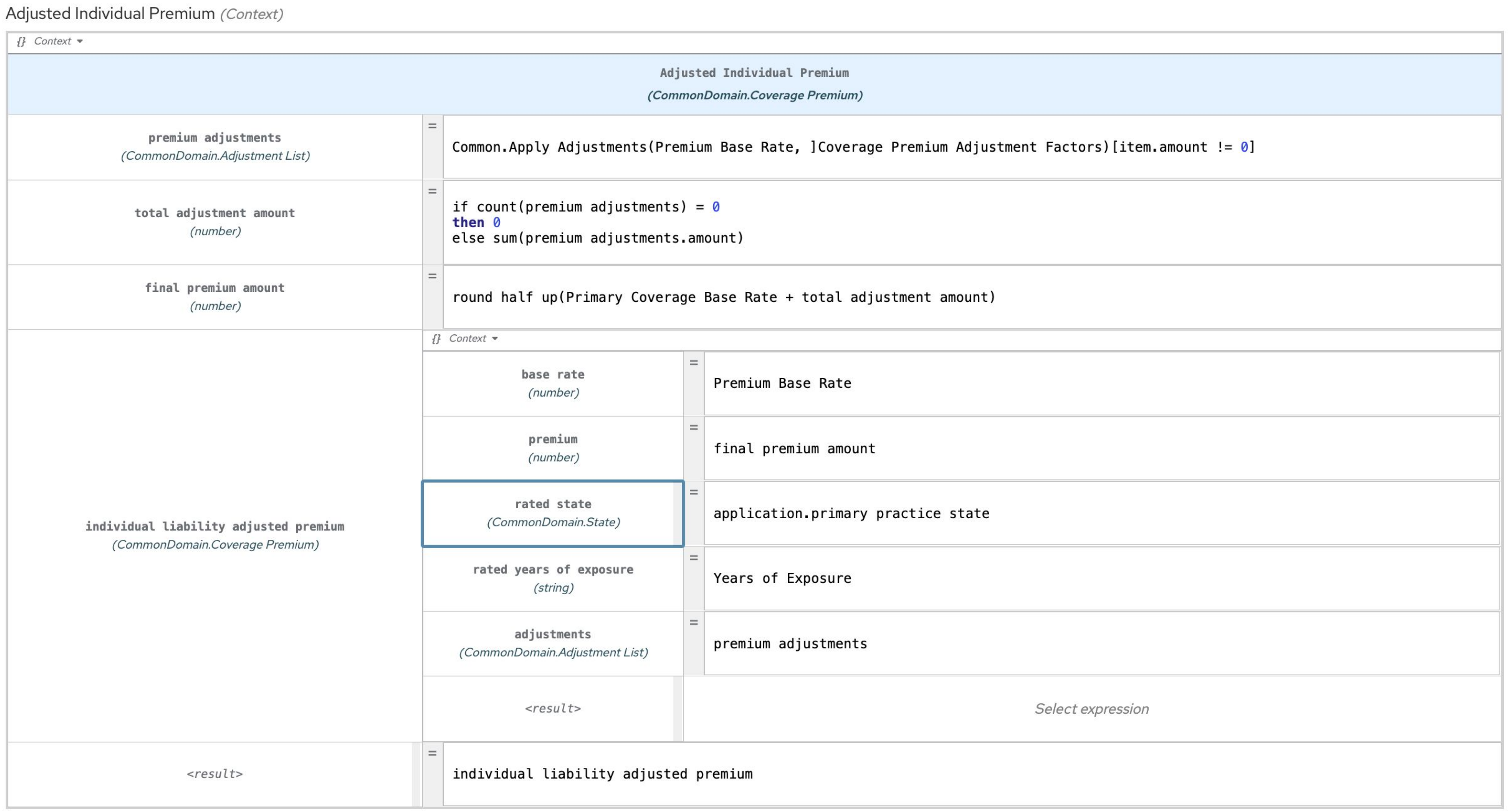
From Requirements to Code


IPL Policy Premium
Common.Calculate Prorate Factor
Policy Term Pro Rate Factor
Minimum Premium
Adjusted Individual Premium
Common.Apply Adjustment
application
Coverage Premium Adjustment Factors
Premium Base Rate
Lookup Primary Coverage Base Rate
Common.Calculate Years of Exposure
Years of Exposure
Vicarious Exposure Amounts
affiliated professionals
Applicable Discounts and Surcharges
Policy Term Pro Rate Factor
Limits of Liability Factor
Claims Free Discount Codes
Common.Calculate Vicarious Exposure
Rate Tables For Rated Territory
Specialty Classification Factor
Physician specialty codes
Practitioner Type Factor
Practitioner Type Factor Codes
Individual Risk Factor
application
base rate tables
Common.Primary Coverage Limit Factor
application
application
Common.Specialty Classification Factor Lookup

There's a faster way to do that.®

AUTOMATING DECISIONS

Both Requirements And Code



Adjusted Individual Premium *(Context)*

{} Context

Adjusted Individual Premium
(CommonDomain.Coverage Premium)

premium adjustments *(CommonDomain.Adjustment List)*	=	`Common.Apply Adjustments(Premium Base Rate, ]Coverage Premium Adjustment Factors)[item.amount != 0]`	

total adjustment amount *(number)*	=	`if count(premium adjustments) = 0`
		`then 0`
		`else sum(premium adjustments.amount)`	

final premium amount *(number)*	=	`round half up(Primary Coverage Base Rate + total adjustment amount)`	
individual liability adjusted premium *(CommonDomain.Coverage Premium)*	{} Context		
	base rate *(number)*	=	`Premium Base Rate`
	premium *(number)*	=	`final premium amount`
	rated state *(CommonDomain.State)*	=	`application.primary practice state`
	rated years of exposure *(string)*	=	`Years of Exposure`
	adjustments *(CommonDomain.Adjustment List)*	=	`premium adjustments`
	<result>		*Select expression*
<result>	=	`individual liability adjusted premium`	

BP3

There's a faster way to do that.®

AUTOMATING DECISIONS

Business Friendly Code

TheQuickBrownFoxJumpedOverTheLazyDog ☹

The_quick_brown_fox_jumped_over_the_lazy_dog ☹

The quick brown fox jumped over the lazy dog 😇

AUTOMATING DECISIONS

There’s a faster way to deliver the right solution

Collaboration with Policy Experts

Sound Testing Strategy

Build & Deployment Pipelines

Scalable Production Environments

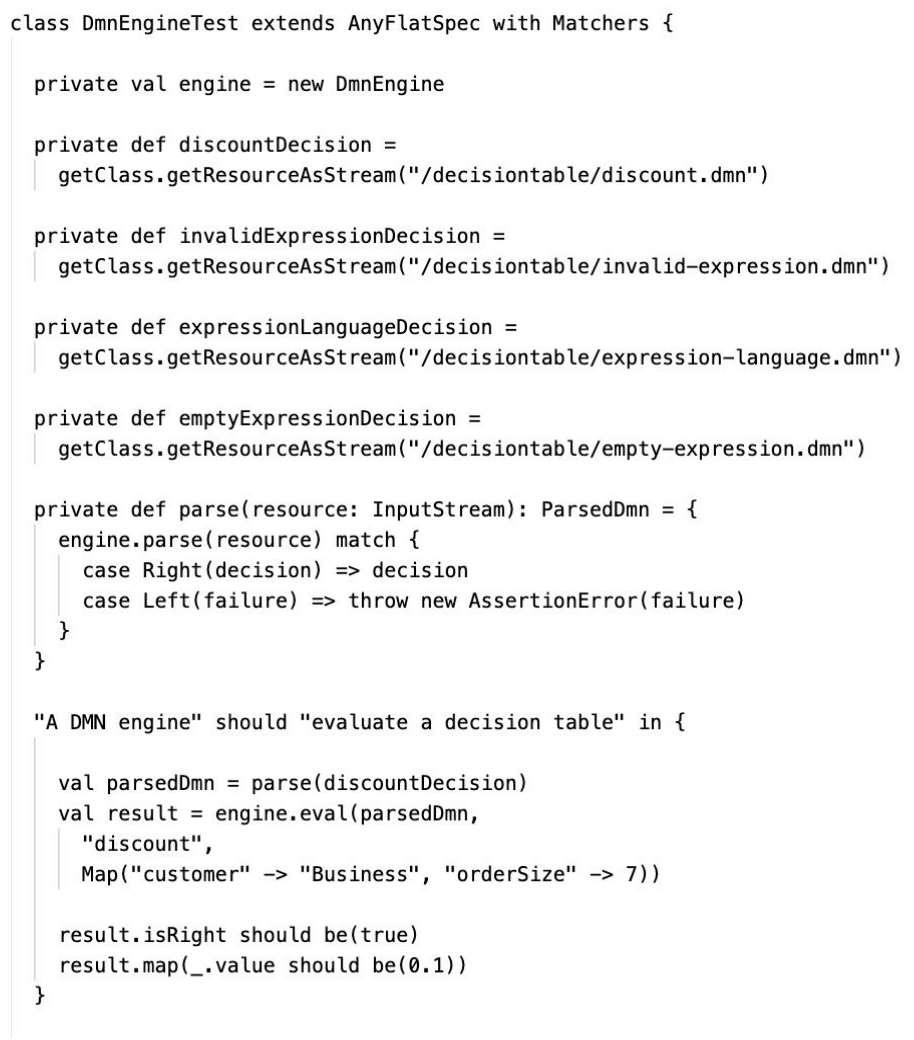
AUTOMATING DECISIONS

Java Tests

```java
public class DmnTckSuite
        extends Suite {

    private static final Logger logger = LoggerFactory.getLogger( DmnTckSuite.class );

    private final Description           descr;
    private final DmnTckVendorTestSuite ntsuite;
    private final List<Runner>          runners;


    public DmnTckSuite(Class<?> clazz)
            throws InitializationError {
        super( clazz, Collections.<Runner>emptyList() );

        runners = new ArrayList<Runner>();

        try {
            ntsuite = (DmnTckVendorTestSuite) clazz.newInstance();
        } catch ( Exception e ) {
            logger.error( "Error instantiating test suite.", e );
            throw new InitializationError( e );
        }
        List<URL> urls = ntsuite.getTestCases();
        this.descr = Description.createSuiteDescription( "DMN TCK test suite" );

        for ( URL url : urls ) {
            File tcFolder = null;
            try {
                tcFolder = new File( url.toURI() );
            } catch ( URISyntaxException e ) {
                throw new InitializationError( e );
```

AUTOMATING DECISIONS

Sound Testing Strategy



```
class DmnEngineTest extends AnyFlatSpec with Matchers {

  private val engine = new DmnEngine

  private def discountDecision =
    getClass.getResourceAsStream("/decisiontable/discount.dmn")

  private def invalidExpressionDecision =
    getClass.getResourceAsStream("/decisiontable/invalid-expression.dmn")

  private def expressionLanguageDecision =
    getClass.getResourceAsStream("/decisiontable/expression-language.dmn")

  private def emptyExpressionDecision =
    getClass.getResourceAsStream("/decisiontable/empty-expression.dmn")

  private def parse(resource: InputStream): ParsedDmn = {
    engine.parse(resource) match {
      case Right(decision) => decision
      case Left(failure) => throw new AssertionError(failure)
    }
  }

  "A DMN engine" should "evaluate a decision table" in {

    val parsedDmn = parse(discountDecision)
    val result = engine.eval(parsedDmn,
      "discount",
      Map("customer" -> "Business", "orderSize" -> 7))

    result.isRight should be(true)
    result.map(_.value should be(0.1))
  }
```

AUTOMATING DECISIONS

Testing with Cucumber

Cucumber Open
Supported by SMARTBEAR

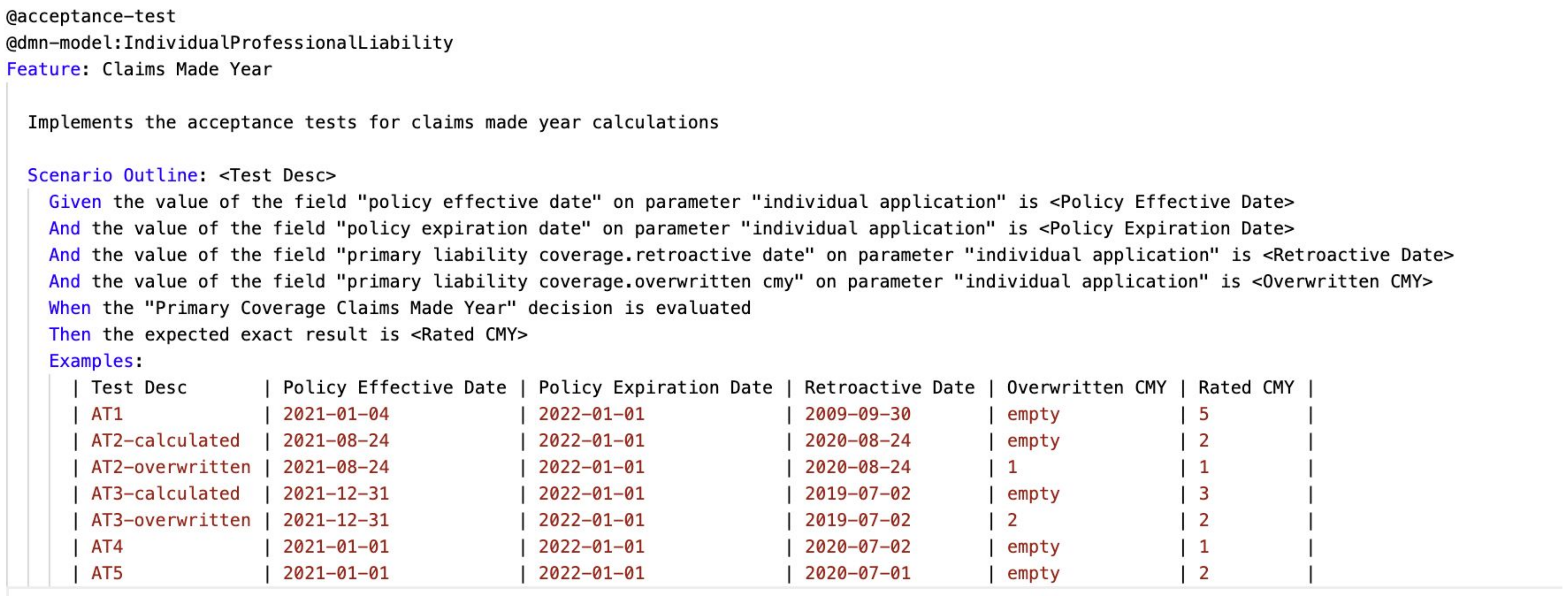
...closes the gap between business people and technical people by:

- Encouraging collaboration across roles…
- Producing system documentation that is automatically checked against the system's behavior

https://cucumber.io/tools/cucumber-open/

AUTOMATING DECISIONS

Testing with Cucumber

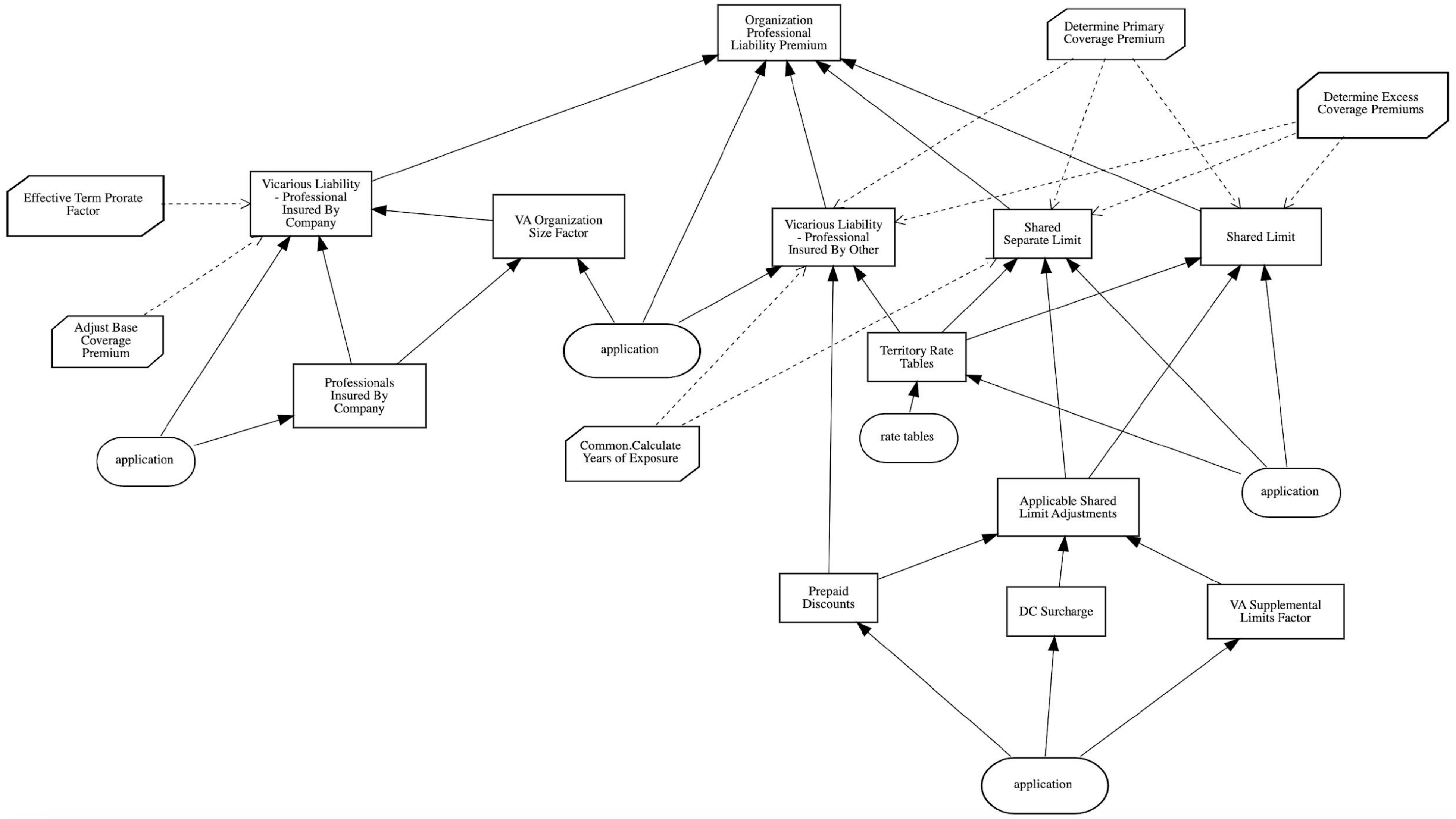


```
@acceptance-test
@dmn-model:IndividualProfessionalLiability
Feature: Claims Made Year

  Implements the acceptance tests for claims made year calculations

  Scenario Outline: <Test Desc>
    Given the value of the field "policy effective date" on parameter "individual application" is <Policy Effective Date>
    And the value of the field "policy expiration date" on parameter "individual application" is <Policy Expiration Date>
    And the value of the field "primary liability coverage.retroactive date" on parameter "individual application" is <Retroactive Date>
    And the value of the field "primary liability coverage.overwritten cmy" on parameter "individual application" is <Overwritten CMY>
    When the "Primary Coverage Claims Made Year" decision is evaluated
    Then the expected exact result is <Rated CMY>
    Examples:
      | Test Desc          | Policy Effective Date | Policy Expiration Date | Retroactive Date | Overwritten CMY | Rated CMY |
      | AT1                | 2021-01-04            | 2022-01-01             | 2009-09-30       | empty           | 5         |
      | AT2-calculated     | 2021-08-24            | 2022-01-01             | 2020-08-24       | empty           | 2         |
      | AT2-overwritten    | 2021-08-24            | 2022-01-01             | 2020-08-24       | 1               | 1         |
      | AT3-calculated     | 2021-12-31            | 2022-01-01             | 2019-07-02       | empty           | 3         |
      | AT3-overwritten    | 2021-12-31            | 2022-01-01             | 2019-07-02       | 2               | 2         |
      | AT4                | 2021-01-01            | 2022-01-01             | 2020-07-02       | empty           | 1         |
      | AT5                | 2021-01-01            | 2022-01-01             | 2020-07-01       | empty           | 2         |
```

There's a faster way to do that.®

AUTOMATING DECISIONS

Re-Using Decision Logic



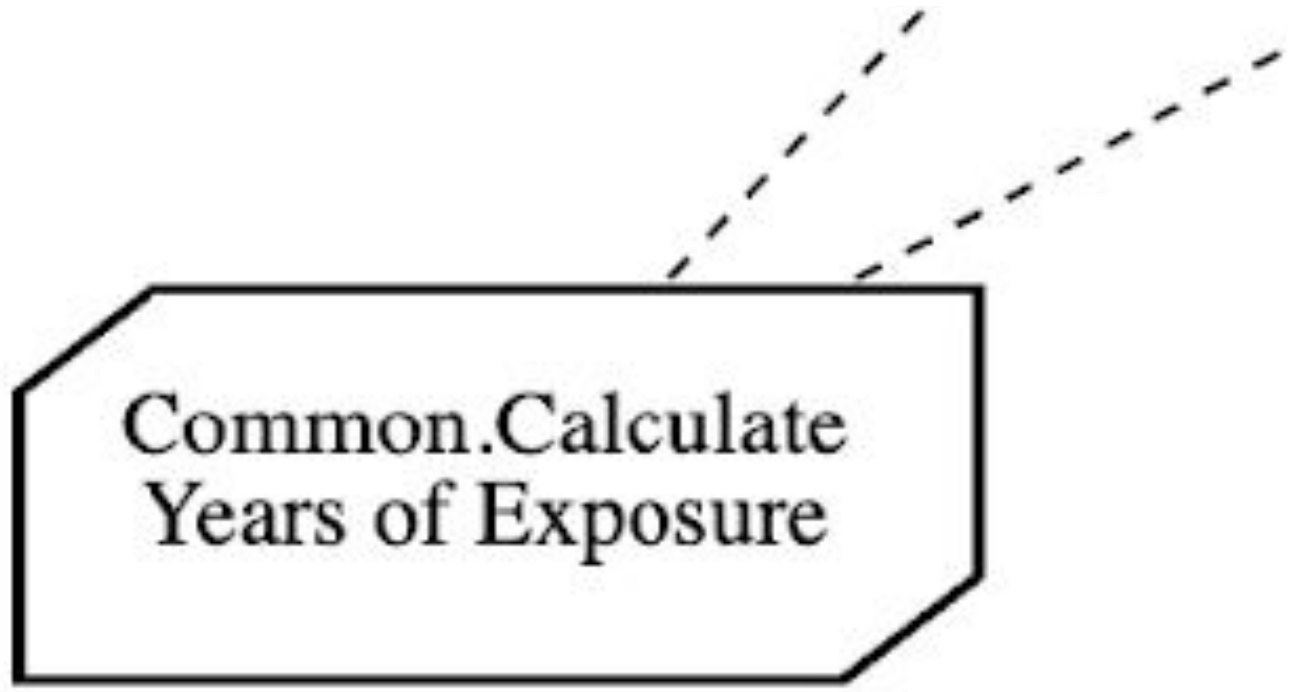
Organization Professional Liability Premium
Determine Primary Coverage Premium
Determine Excess Coverage Premiums
Effective Term Prorate Factor
Vicarious Liability - Professional Insured By Company
VA Organization Size Factor
Vicarious Liability - Professional Insured By Other
Shared Separate Limit
Shared Limit
Adjust Base Coverage Premium
application
Territory Rate Tables
Professionals Insured By Company
rate tables
application
Common.Calculate Years of Exposure
application
Applicable Shared Limit Adjustments
Prepaid Discounts
DC Surcharge
VA Supplemental Limits Factor
application

BP3

There's a faster way to do that.®

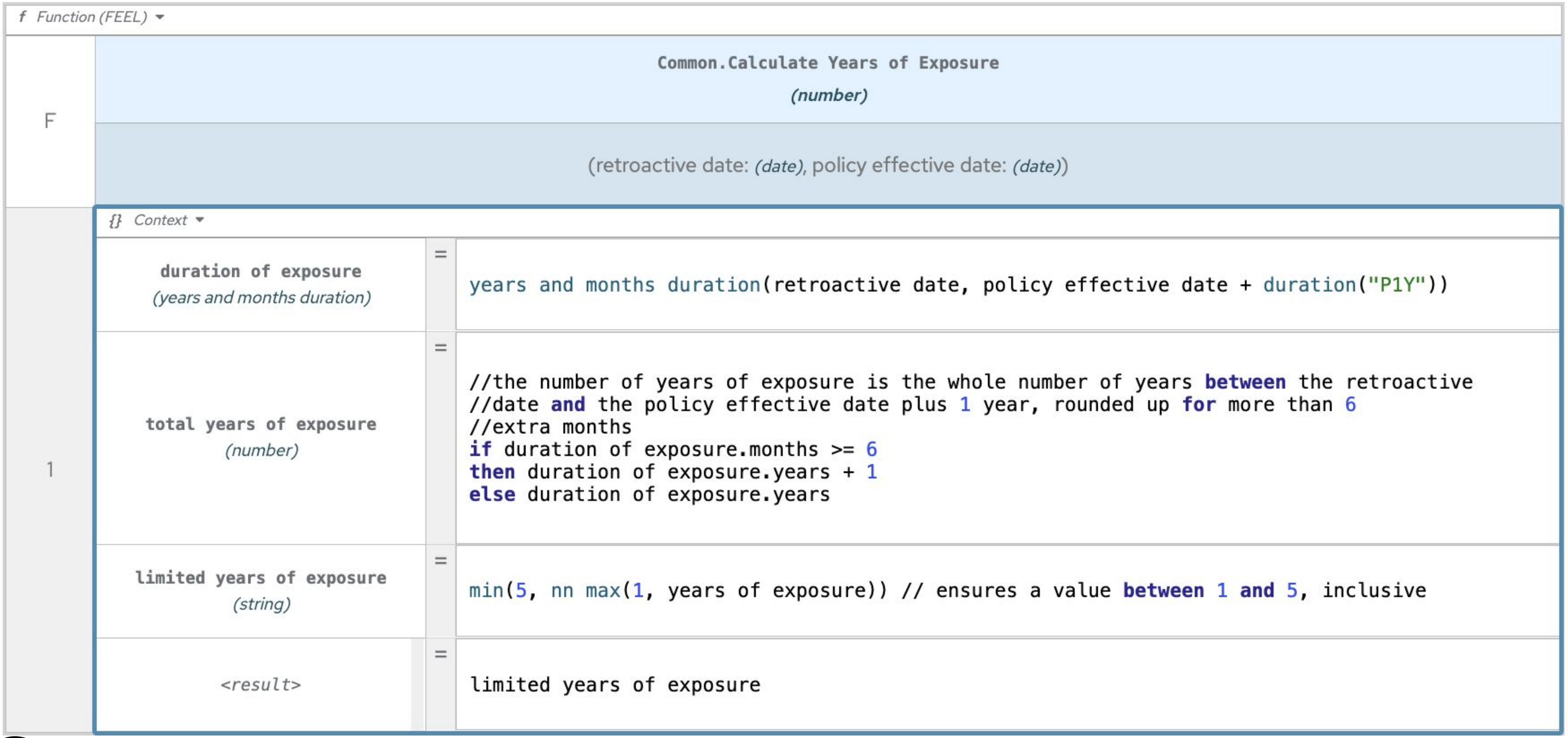
AUTOMATING DECISIONS

Re-Using Decision Logic



AUTOMATING DECISIONS

Re-Using Decision Logic



f Function (FEEL)

F

Common.Calculate Years of Exposure
(number)

(retroactive date: *(date)*, policy effective date: *(date)*)

1

{} Context

duration of exposure *(years and months duration)*	`years and months duration(retroactive date, policy effective date + duration("P1Y"))`
total years of exposure *(number)*	see below
limited years of exposure *(string)*	`min(5, nn max(1, years of exposure)) // ensures a value between 1 and 5, inclusive`
<result>	`limited years of exposure`

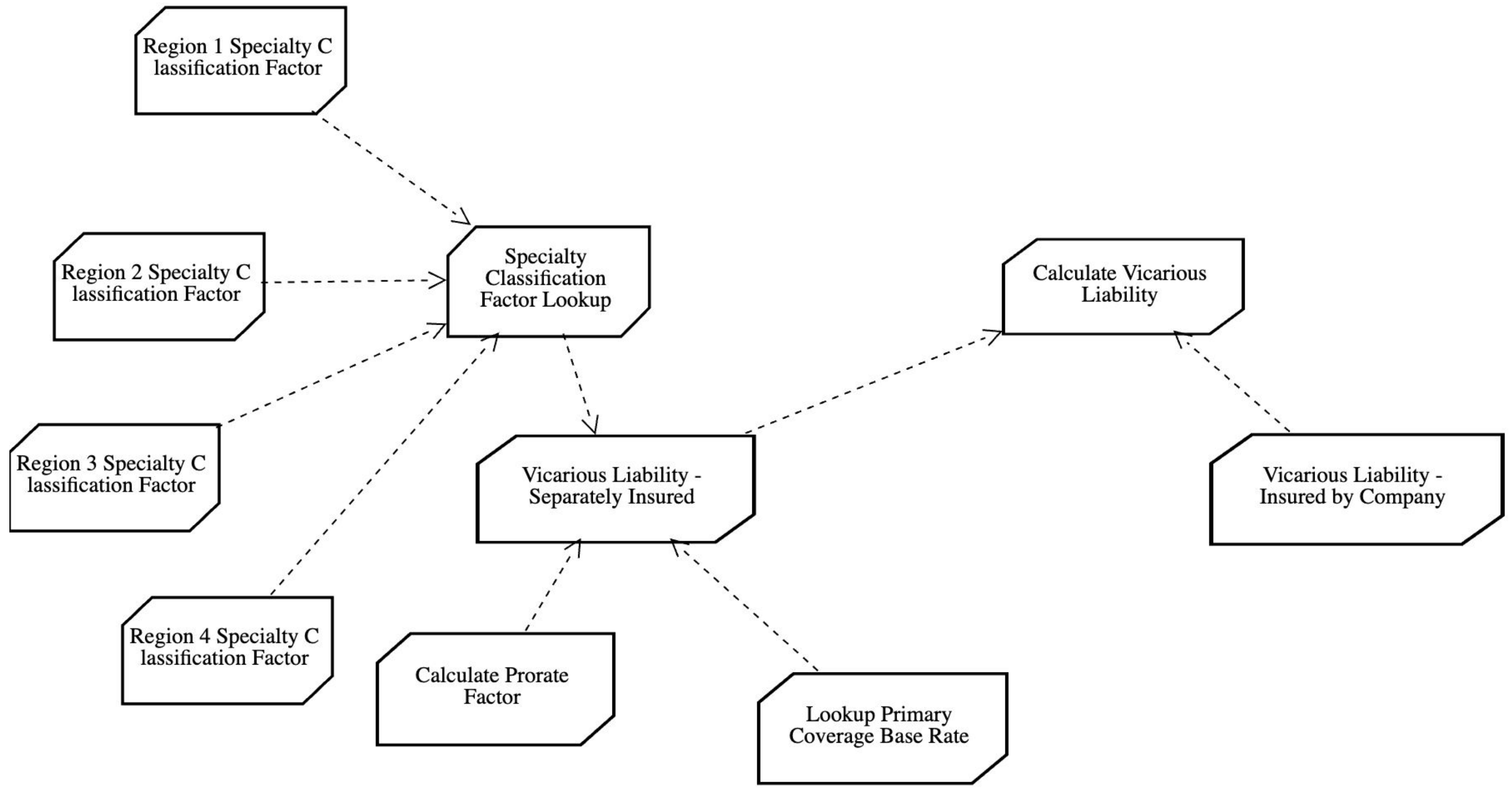
total years of exposure =

```
//the number of years of exposure is the whole number of years between the retroactive
//date and the policy effective date plus 1 year, rounded up for more than 6
//extra months
if duration of exposure.months >= 6
then duration of exposure.years + 1
else duration of exposure.years
```

There's a faster way to do that.®

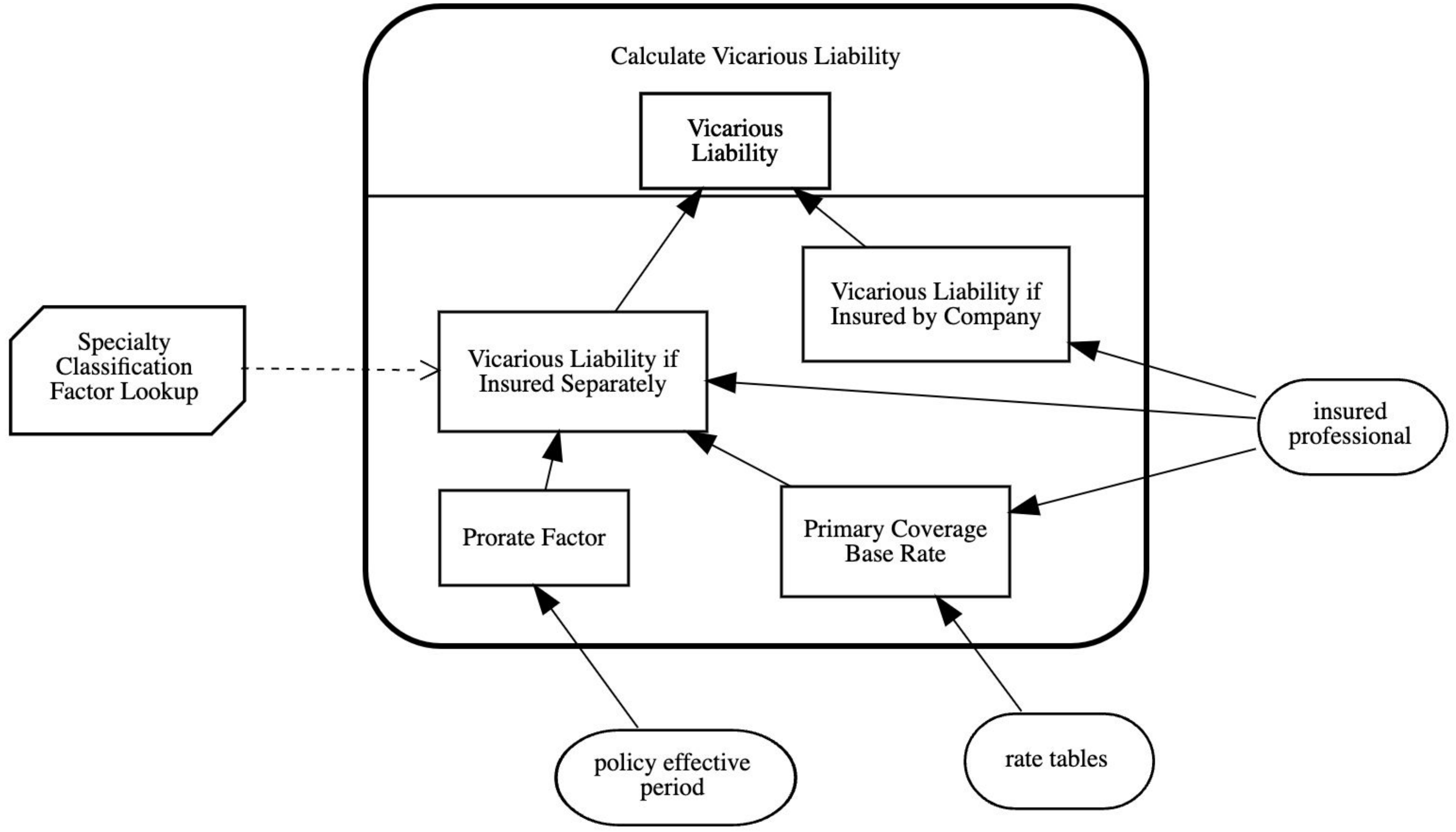
DMN EXECUTION ENGINES

From Red Hat to Camunda 8 – Next Steps


Region 1 Specialty C lassification Factor
Region 2 Specialty C lassification Factor
Region 3 Specialty C lassification Factor
Region 4 Specialty C lassification Factor
Specialty Classification Factor Lookup
Vicarious Liability - Separately Insured
Calculate Prorate Factor
Lookup Primary Coverage Base Rate
Calculate Vicarious Liability
Vicarious Liability - Insured by Company

DMN EXECUTION ENGINES

From Red Hat to Camunda 8 – Next Steps


Calculate Vicarious Liability
Vicarious Liability
Vicarious Liability if Insured by Company
Specialty Classification Factor Lookup
Vicarious Liability if Insured Separately
insured professional
Prorate Factor
Primary Coverage Base Rate
policy effective period
rate tables

BP3

There's a faster way to do that.®

DMN EXECUTION ENGINES

From Red Hat to Camunda 8 – Next Steps

- Finish the Work on Included Models
- Add Support for Decision Services