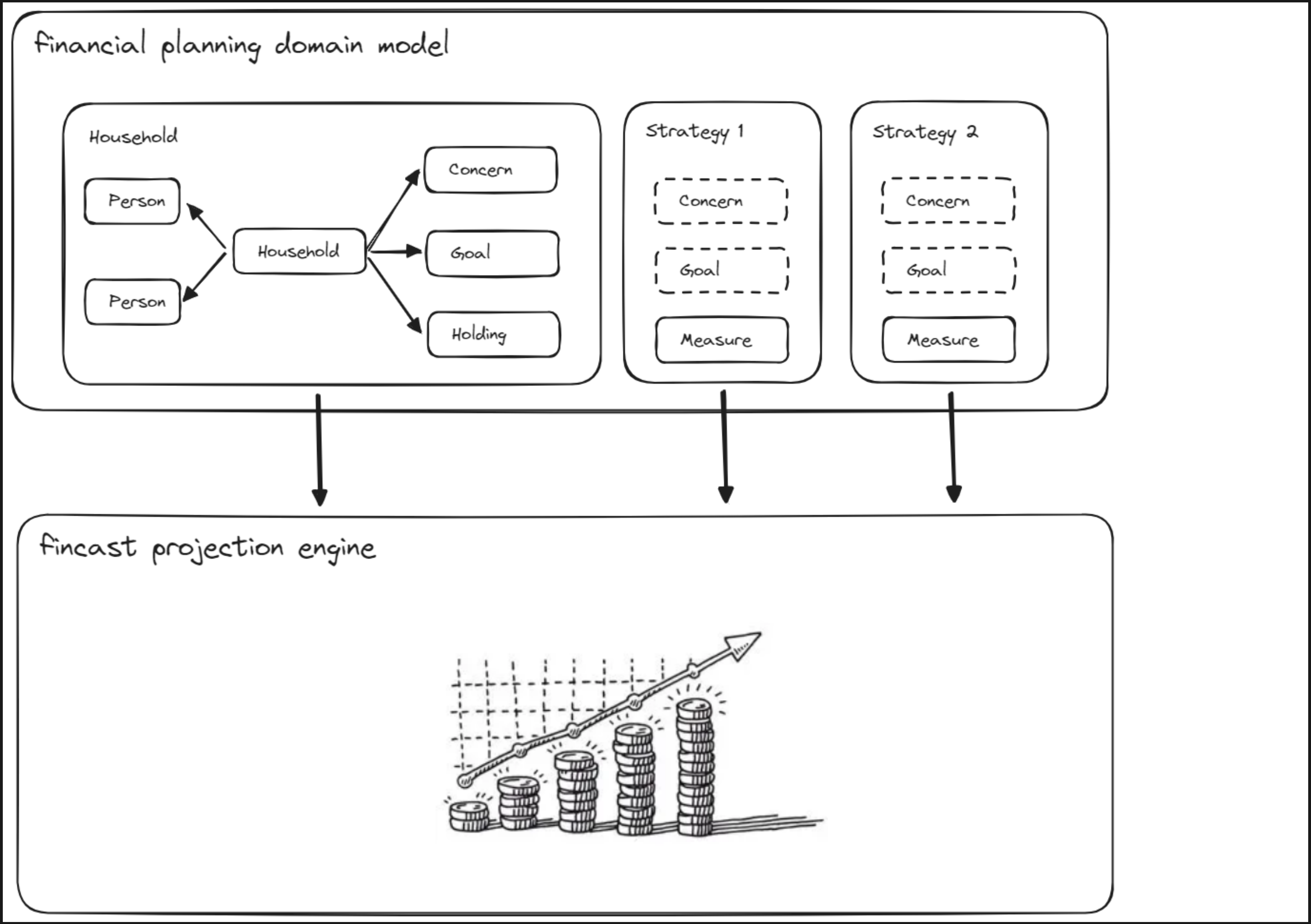
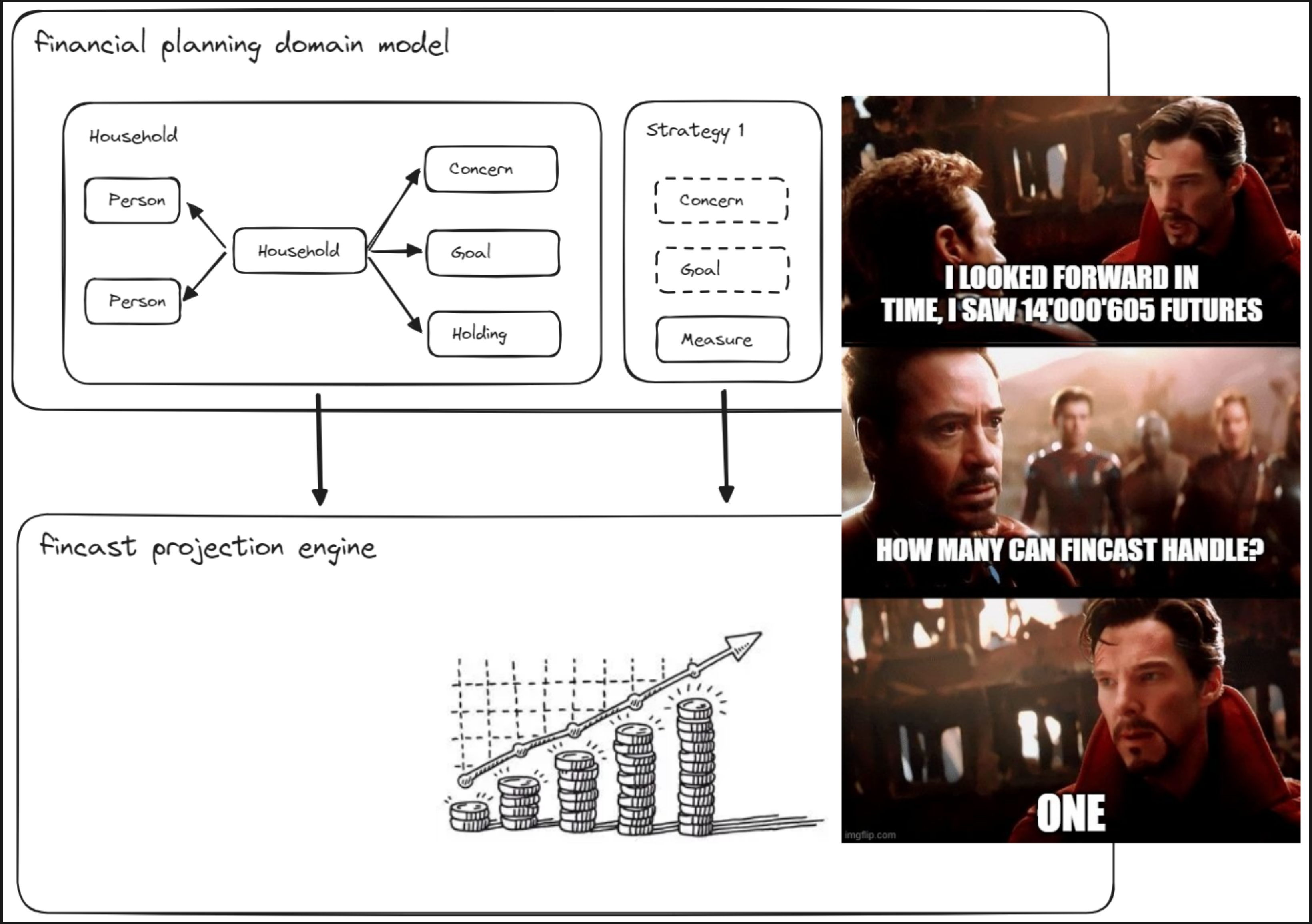
FINCAST CONCEPTS

DFP 2024

DOMAIN MODEL VS FINCAST



DOMAIN MODEL VS FINCAST


financial planning domain model
Household
Person
Person
Household
Concern
Goal
Holding
Strategy 1
Concern
Goal
Measure
fincast projection engine
I LOOKED FORWARD IN TIME, I SAW 14'000'605 FUTURES
HOW MANY CAN FINCAST HANDLE?
ONE
imgflip.com

domain model:

- model of **all** potential futures
- holdings, bookings, goals, concerns, scenarios, measures, ...

fincast:

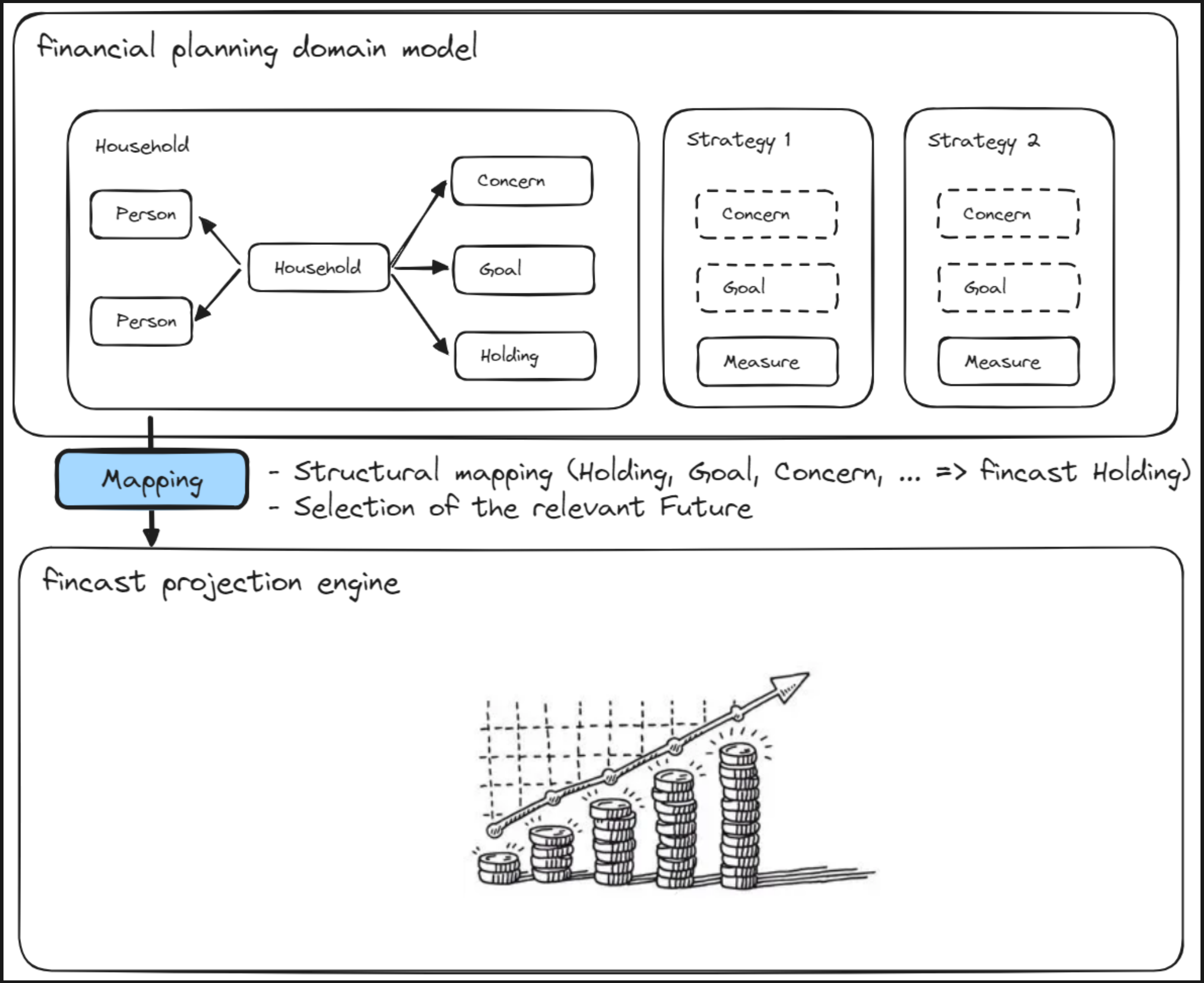
- model of **exactly one** future
- household, holdings, bookings

domain model	fincast
goal: potential early retirement	retire early at 63
goal: buy a yacht in around 4 years, for about 200k	buy a yacht in 2028 for 150k
concern: health care costs	dental bill of 25k in 2027
measure: phased payout pillars 3a	in 2036 payout of 2nd 3a account

MAPPING

- fincast does not need to know all the complexities of the domain model, neither **structurally** nor **optionalities**
- therefore, we need a **mapping** from domain model to fincast
- instead of building a complex metadata model we accept explicitly coded mapping as a tradeoff
- fincast product model can be kept rather simple and generic since we are mapping anyway

MAPPING

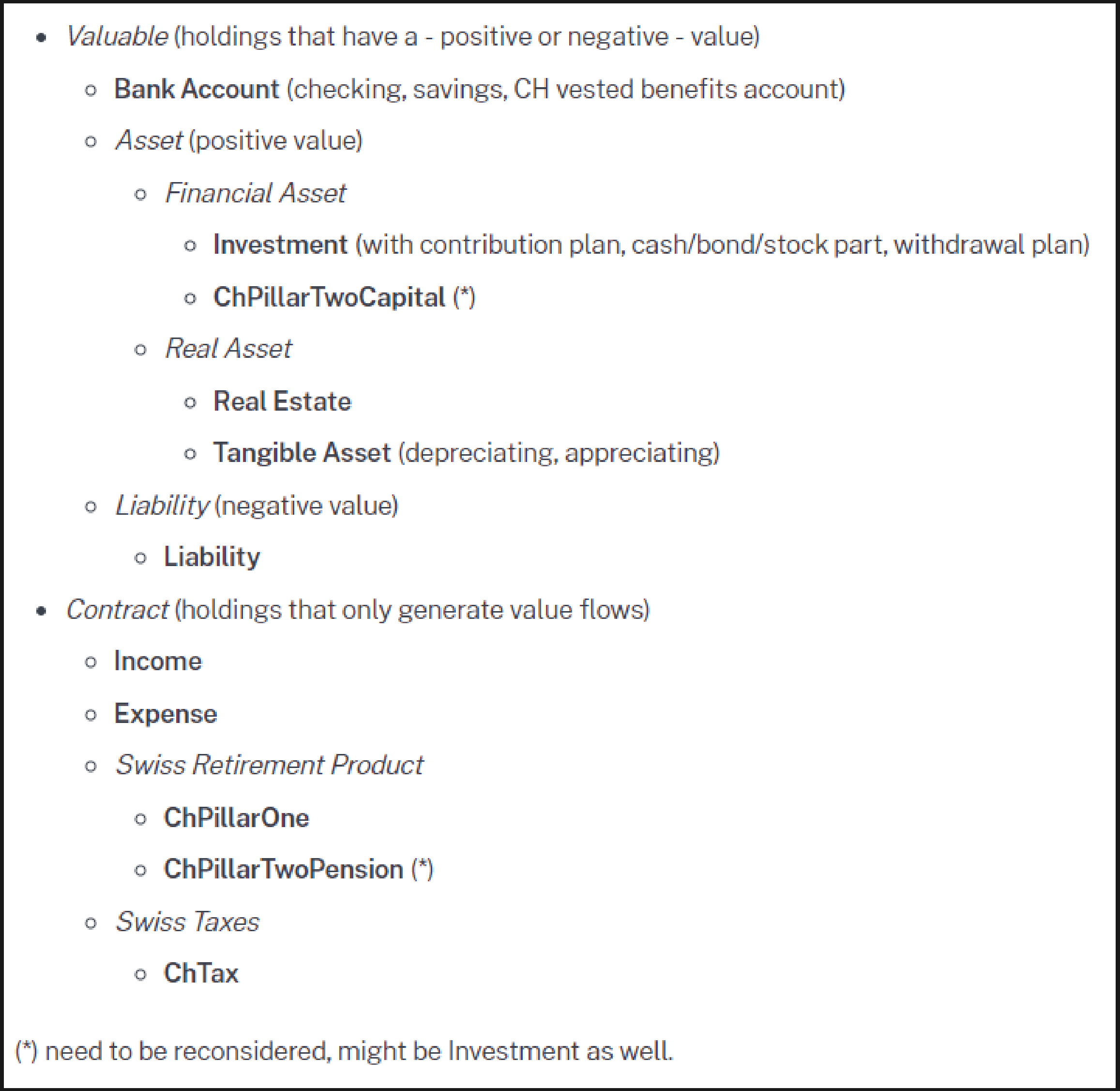


- Structural mapping (Holding, Goal, Concern, ... => fincast Holding)
- Selection of the relevant Future

HOUSEHOLD STRUCTURE (FINCAST)

- Household
- Person (partner1 and optional partner2)
- Dependent (children) [nyi]
- Holding (no product entity)
- Booking (discretionary bookings, mainly through goals and measures) [nyi]

PRODUCTS / HOLDINGS



- *Valuable* (holdings that have a - positive or negative - value)
    - **Bank Account** (checking, savings, CH vested benefits account)
    - *Asset* (positive value)
        - *Financial Asset*
            - **Investment** (with contribution plan, cash/bond/stock part, withdrawal plan)
            - **ChPillarTwoCapital** (*)
        - *Real Asset*
            - **Real Estate**
            - **Tangible Asset** (depreciating, appreciating)
    - *Liability* (negative value)
        - **Liability**
- *Contract* (holdings that only generate value flows)
    - **Income**
    - **Expense**
    - *Swiss Retirement Product*
        - **ChPillarOne**
        - **ChPillarTwoPension** (*)
    - *Swiss Taxes*
        - **ChTax**

(*) need to be reconsidered, might be Investment as well.

HOLDING GRANULARITY

The holding granularity for projection is a mixture of what is mandated by fincast and domain model convention.

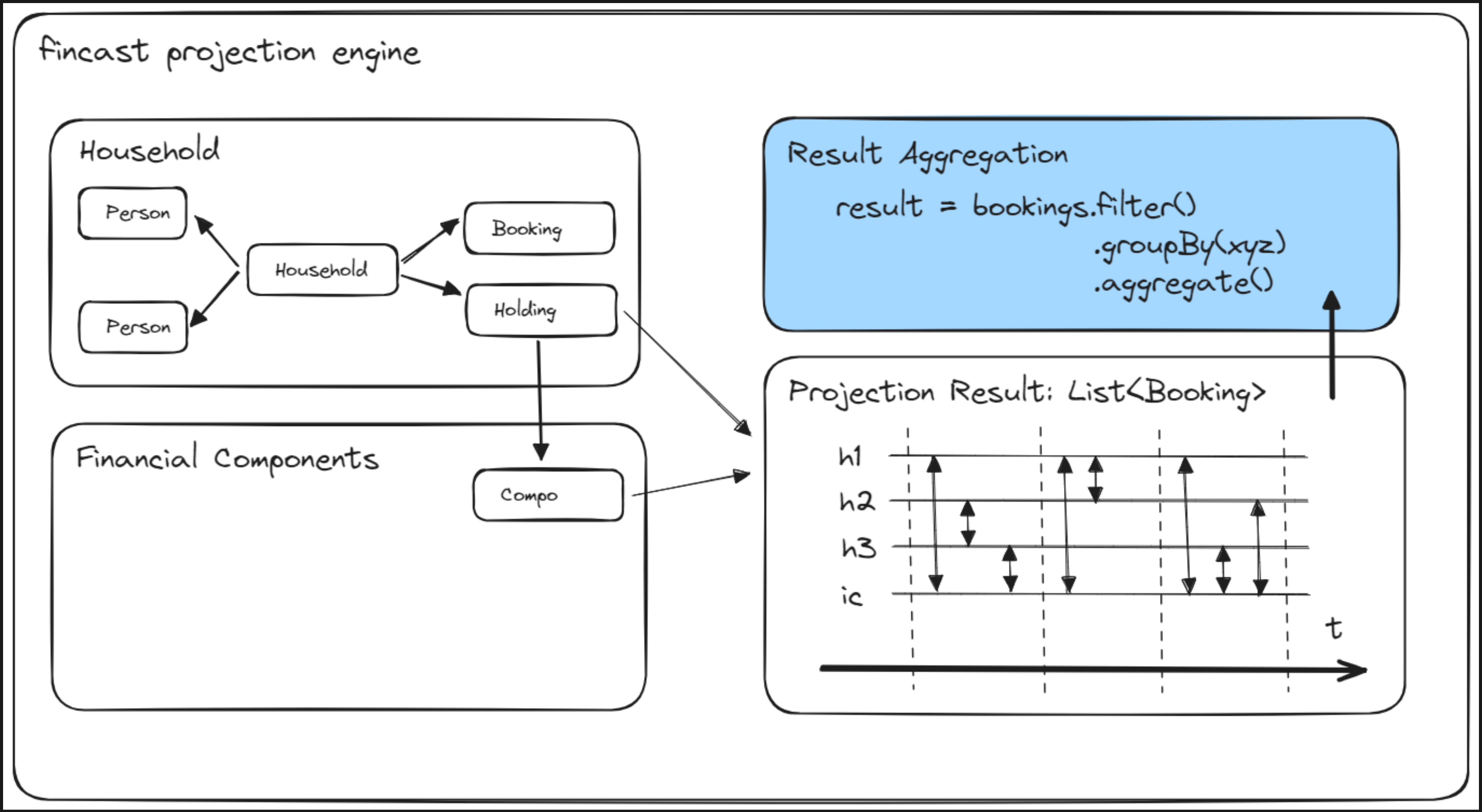
DEDICATED HOLDINGS

- regulated, personal retirement provision products (AHV, BVG, 1e, 3a) with individual cashflow schemes
- large and/or illiquid assets (real estate) or liabilities (mortgage)
- **bucket Holdings**: holdings that for various (client mandated) reasons need to be tracked separately. typically linked to a specific goal, concern, measure.

SPECIAL (SUMMARY) HOLDINGS

- **external Cash**: placeholder for all external cash holdings
- **internal Cash**: aggregation of non-bucket bank accounts
- **buffer Investment**: analoguos to internal cash, sum of all non-bucket investments (currently only one). they are used to buffer excess cash: they cover negative cashflow but also are topped up through positive cashflows.

PROJECTION ENGINE



```
result = bookings.filter()
                 .groupBy(xyz)
                 .aggregate()
```

PROJECTION RESULT: BOOKINGS

- the result of a projection run is a list of half-bookings, with complete information about counter and triggering holding
- for example, a salary will trigger monthly bookings from externalCash to internalCash with trigger salary holding

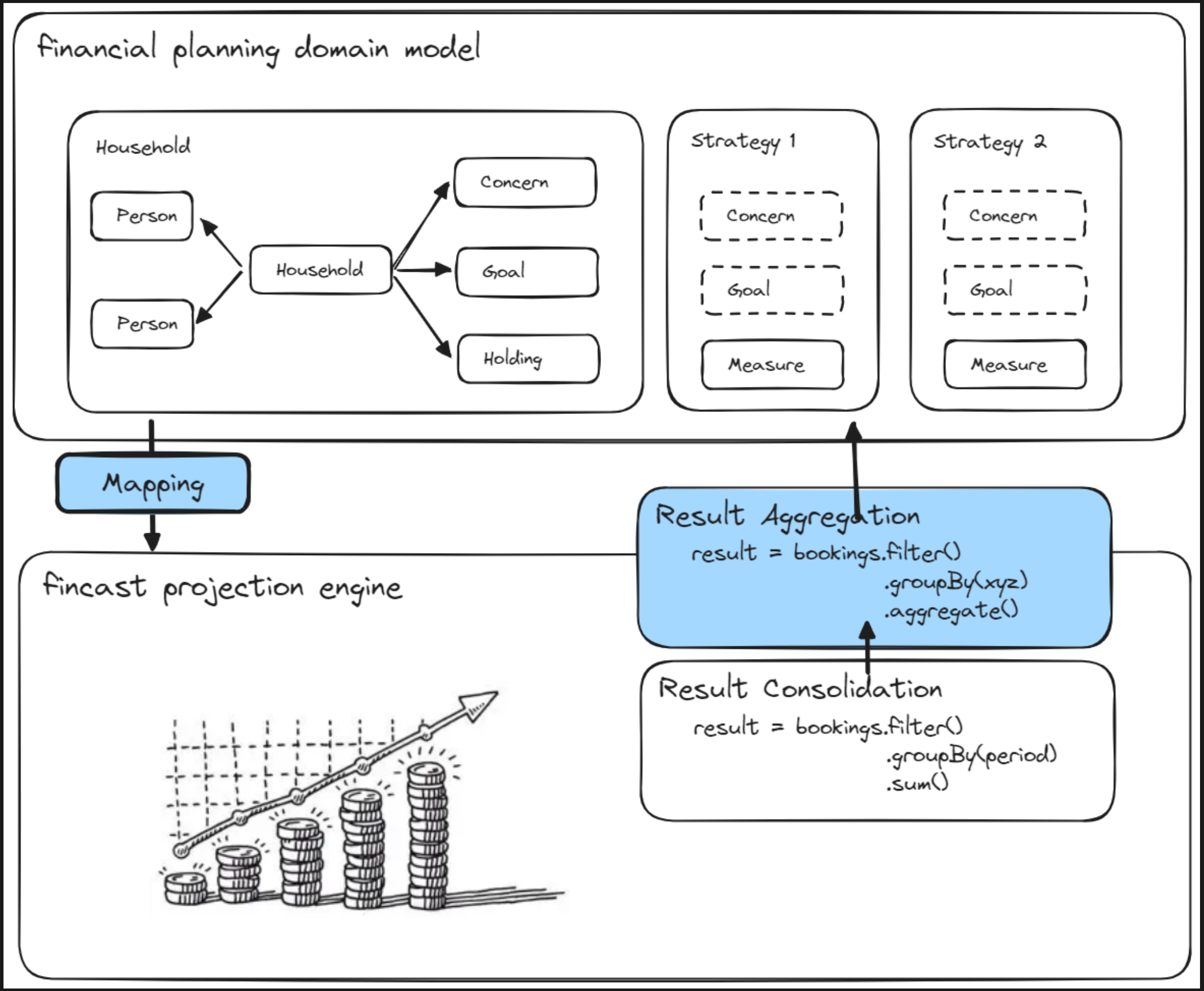
```
val holding: Holding,
val date: SimDate,
val bookingKind: BookingKind,
val amount: Double,
val counterHolding: Holding,
val trigHolding: Holding,
```

CONSOLIDATION & AGGREGATION

The aggregation process is split into 2 phases:

- Consolidation (fincast)
- Aggregation (domain model)

MAPPING, CONSOLIDATION & AGGREGATION



```
result = bookings.filter()
                 .groupBy(xyz)
                 .aggregate()
```

```
result = bookings.filter()
                 .groupBy(period)
                 .sum()
```

CONSOLIDATION

Periodic (monthly/yearly) compression:

- **booking consolidation**
- **holding consolidation**
- **portfolio consolidation**

in this step, given a "portfolio" (= list of holdings), it will also calculate inflows and outflows on portfolio level, i.e. netting them on portfolio level. this portolio needs to be defined from the "consumer" (domain model), f.ex. to include / exclude pension products.

HOLDING CONSOLIDATION

```kotlin
data class HoldingPeriod(
    val date: SimDate,
    val holding: Holding,
    var balance: Double,
    var inflows: Double,
    var outflows: Double,
    var gain: Double,
    val bookings: List<Booking>,
)
```

PORTFOLIO CONSOLIDATION

```
data class PortfolioPeriod(
    val date: SimDate,
    var assets: Double,
    var liabilities: Double,
    var inflows: Double,
    var outflows: Double,
    var gain: Double,
    val bookings: List<Booking>,
)
```

AGGREGATION

- since the preliminary result of the projection engine is still a monthly list of consolidated half-bookings, we need to aggregate them first to get a useful result
- aggregation rules are typically defined in domain model concepts, which are not necessarily known to fincast engine (f.ex. group by "liquidityClass")
- therefore aggregation is not part of the core engine (but likely can be specified from the consumer through exit points)

AGGREGATION FOR REPORTS

Roughly speaking, a report is a function that

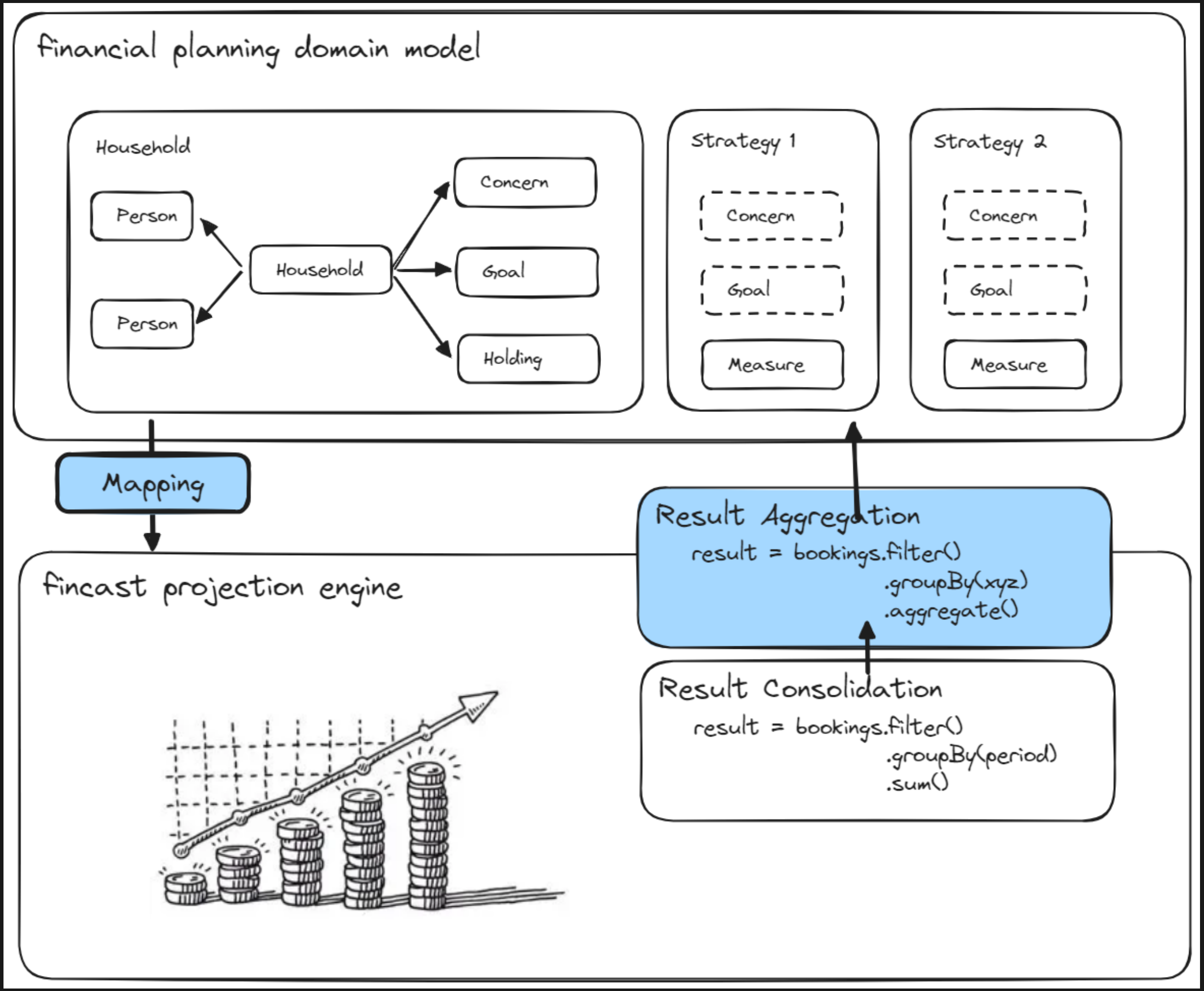
- takes the list of all bookings
- filters them (f.ex. exclude pension products)
- groups them (f.ex. group by liquidityClass)
- aggregates them (f.ex. typically sum)

```
val bookings: List<Booking>
val result: List<Period> =
      bookings.filter { !it.holding.product.isPensionProduct}
              .groupBy { it.holding.product.liquidityClass }
              .mapValues { it.value.sumByDouble { it.amount } }
```

AGGREGATION FOR REPORTS

Of course, we cannot report on more detail than we have in the bookings, so we need to make sure that the bookings are detailed enough to support the reports we want to generate.

MAPPING, CONSOLIDATION & AGGREGATION



Result Aggregation

```
result = bookings.filter()
                 .groupBy(xyz)
                 .aggregate()
```

Result Consolidation

```
result = bookings.filter()
                 .groupBy(period)
                 .sum()
```

PROJECTION LISTENERS

- the engine offers its results, the bookings, through listener interface `ProjectionListener`
- when the "booking engine" creates a new booking, it will notify all registered booking listeners.

```
interface ProjectionListener {
    fun onStartOfMonth(date: SimDate) {}
    fun onBooking(booking: Booking) {}
    fun onEndOfMonth(date: SimDate) {}
}
```

TAXES

How can we differentiate between an investment account and a pillar 3a account for tax purposes?

```
Holding.taxCodes: List<String>
```

TAXES

ChTax holding is also a `ProjectionListener`:

```
// Income Tax Aggregation
override fun onBooking(booking: Booking) {
    ...
    // DEDUCTABLE EXPENSES
    if (holding == household.externalCash && amount > 0) {
        if (trigHolding is RealEstate) {
            taxAggregation.hhBuildingMaintenanceCosts += amount
        } else if (trigHolding is Expense && trigHolding.hasTaxCode(
            taxAggregation.hhRentalExpense += amount
        } else if (bookingKind == BookingKind.INTEREST) {
            taxAggregation.hhDebtInterest += amount
        }
    }
    ...
}
```

TAXES

Current list of tax codes:

```
val SalaryPrimary // ESTV detail, triggers primary deductions
val SalarySecondary // ESTV detail, triggers secondary deductions
val ExpenseRent // deduction
val ExpenseAlimonyMinor // deduction
val ExpenseAlimonyAdult // deduction
val Investment3a // deduction
```

YEARLY REBALANCING

Let's have a look at the projection lifecycle phases:

1. Monthly Holding Lifecycle (cashflow, interest, dividends, ...)
2. Discretionary Bookings (f.ex. "buy yacht in June 2028 for 150k") [nyi]
3. EOM Transfers of Holdings (f.ex. "transfer 100% of 3a account to internal cash")
4. EOY Taxes
5. EOY Rebalancing

PROPOSED EOY REBALANCING ALGORITHM

1. process **discretionary, unconditional** flows / bookings (f.ex. "transfer the yearly max amount to my current 3a account")
2. determine whether we have a positive or negative free cashflow in the year at this point
3. **distribute positive free cashflow** according to rules (see below), funded from internalCash
4. **cover negative cashflow** from **buffer Investment** (try to bring internalCash to target balance)

DISTRIBUTION OF POSITIVE FREE CASHFLOW

If there is a **positive free cashflow** in the year (when we exceed the **target cash balance** on internalCash), we can distribute to other holdings according to rules (sequentially, in an absolute or relative way), f.ex.:

1. 10% of fcf, capped by yearly max, to my 3a account
2. CHF 5'000 to my Whisky collection
3. 20% of fcf to my investment bucket XYZ
4. rest, if any, go to buffer investment

BOOKINGS / FLOWS

Note that this model supports 4 levels of bookings

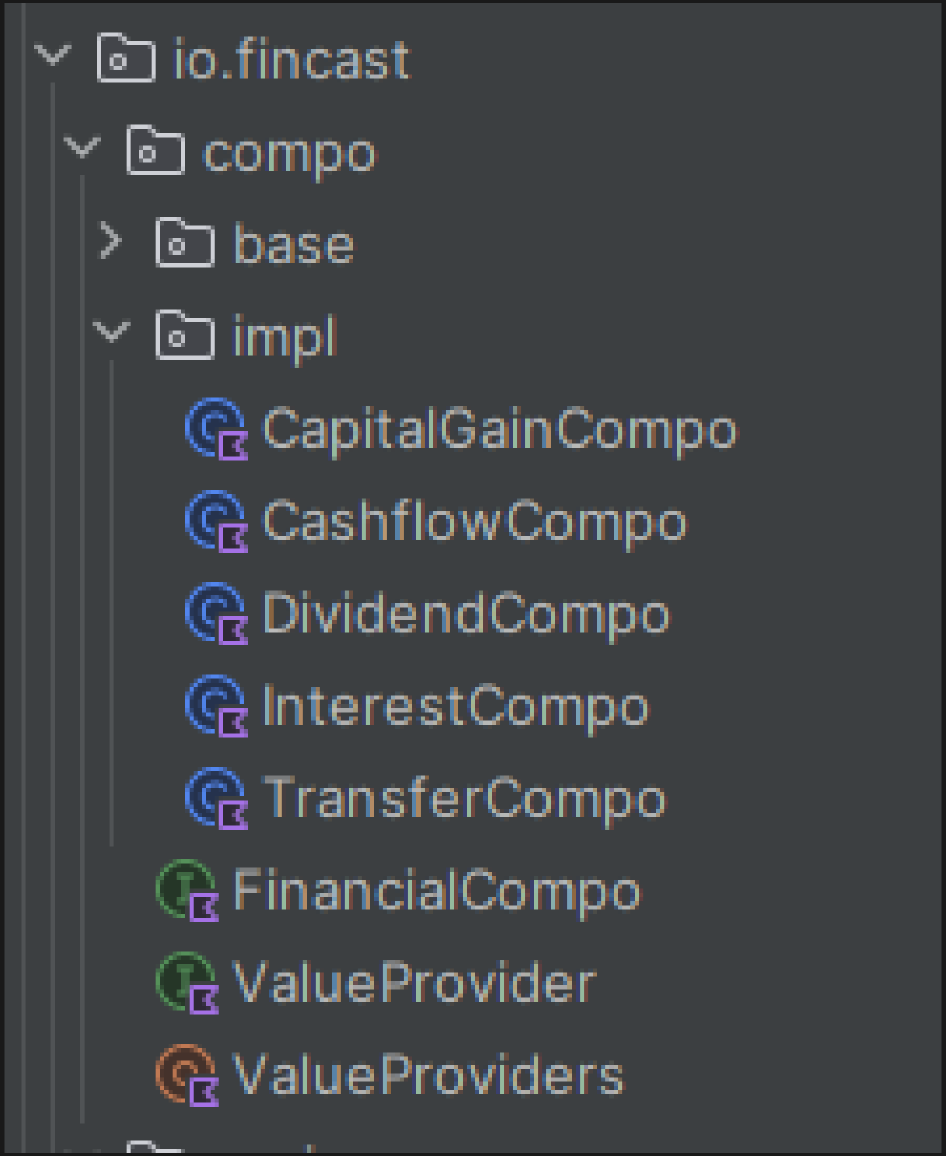
- Contractual flows can be modeled through the holding, f.ex. "transfer yearly max to my **insurance pillar 3a**" (which enforces payments)
- Single, discretionary, unconditional, flows can be modeled through **"discretionary bookings"**
- Repeating, unconditional flows can be modeled through **"unconditional rebalancing bookings"**
- Discretionary, conditional flows can be modeled through **"conditional rebalancing bookings"** (subject to available free cashflow)

VARIOUS TOPICS

TIME AND MONEY RESOLUTION

- **SimDate** provides for a monthly time resolution
- Monetary amounts are kept as **Double**, we are not keeping books to the cent

FINANCIAL COMPONENTS



```
io.fincast
  compo
    base
    impl
      CapitalGainCompo
      CashflowCompo
      DividendCompo
      InterestCompo
      TransferCompo
    FinancialCompo
    ValueProvider
    ValueProviders
```

CONFIGURATION: KOTLIN DSL

Termsheet

```
Termsheet(tme, "real_estate") {
  name = "Real estate"
  term("fiscalValue") { type = "value"; name = "Fiscal Value" }
  term("imputedRentalValue") { type = "value"; name = "Imputed Rental Value
  term("isSelfOccupied") { type = "boolean"; name = "Is self occupied?" }
  term("rentalIncome") { type = "value"; name = "Rental Income (yearly)" }
  term("maintenanceRate") { type = "percentage"; name = "Maintenance Rate"
}
```

CONFIGURATION: KOTLIN DSL

Product

```kotlin
Product("ch_pillar_3a_account") {
  name = "Säule 3a Konto"
  description = "Säule 3a Konto"
  productModel = "investment"
  liquidityClass = "retirement"
  term("contributionPeriodicity") { value = "yearly"; +frozen; +hidden }
  term("cash") { value = "100"; +hidden }
  term("cashInterestRate") { value = "2" }
  term("bond") { value = "0"; +hidden }
  term("bondInterestRate") { +hidden }
  term("stock") { value = "0"; +hidden }
  term("stockDividendYield") { +hidden }
  term("stockCapitalGain") { +hidden }
  term("managementFee") { value = "0"; +hidden }
  term("performanceFee") { value = "0"; +hidden }
}
```

MATRIMONIAL REGIME

One of the challenges in matrimonial regimes is the calculation of the "Eigengut". This can be modelled as an asset and changes can be tracked.

```
// Eigengut Aggregation (ProjectionListener)
override fun onBooking(booking: Booking) {
    ...
    // INCOMING INHERITANCE & GIFTS (modelling tbd)
    if (amount > 0 && "booking is an inheritance" && holding.owner =
        holding.value += amount
    } else if (amount > 0 && "booking is a Gift"&& holding.owner ==
        holding.value += amount
    }
    ...
}
```

WORK IN PROGRESS

- Domain Model with Goals, Concerns, Strategies, Measures, ...
- Inflation
- Consolidation & Aggregation (technical)
- Rebalancing (subject to BAs)
- Products in Detail (subject to BAs)

FIN