Programmierer sind von Natur aus inkompetent

Michael Sperber

Bewertung von Finanzderivaten

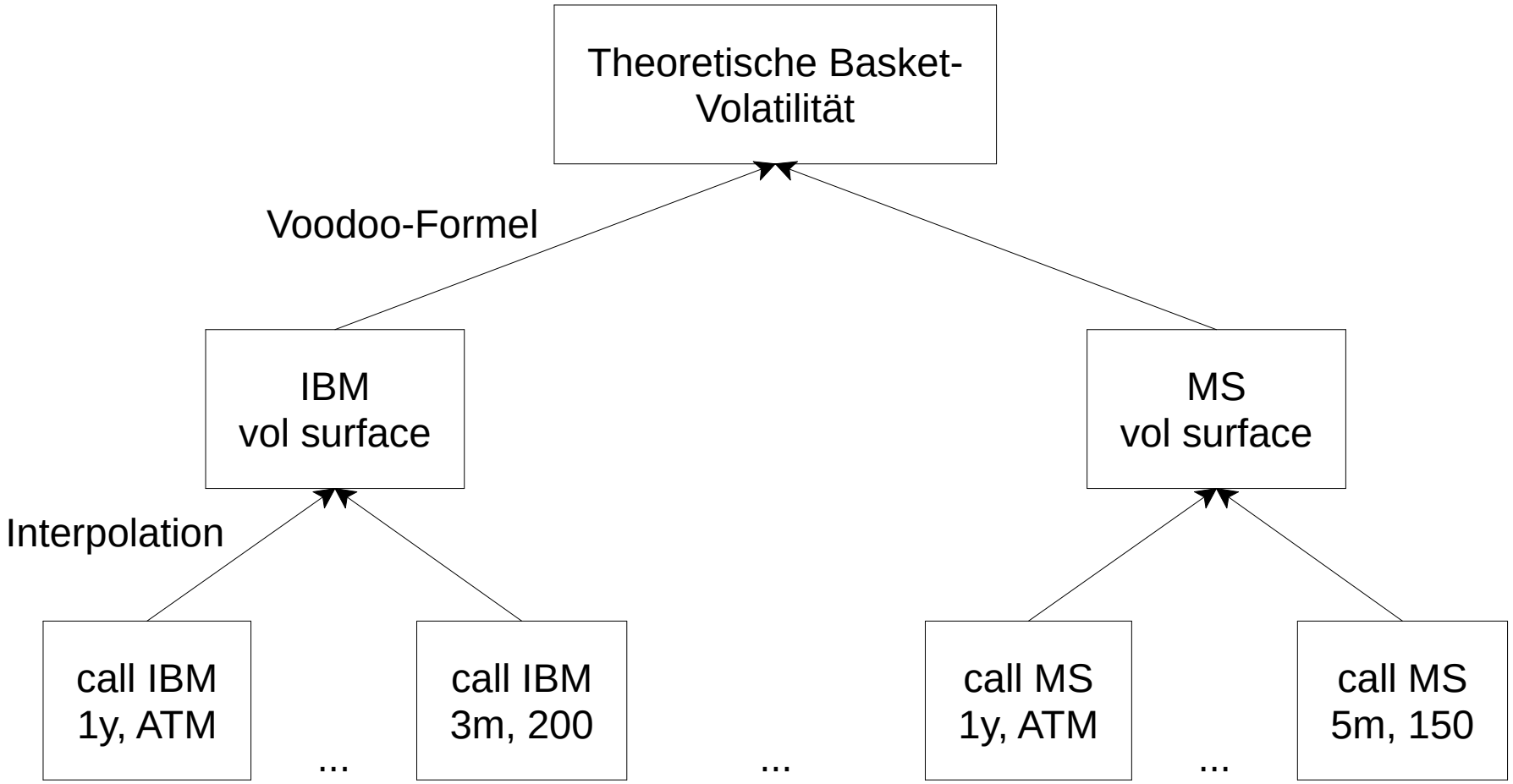
$$p = f(d, M)$$

- d ist ein *Finanzderivat*
- p ist ein *theoretischer Preis*
- M ist eine Quelle für *Marktdaten*

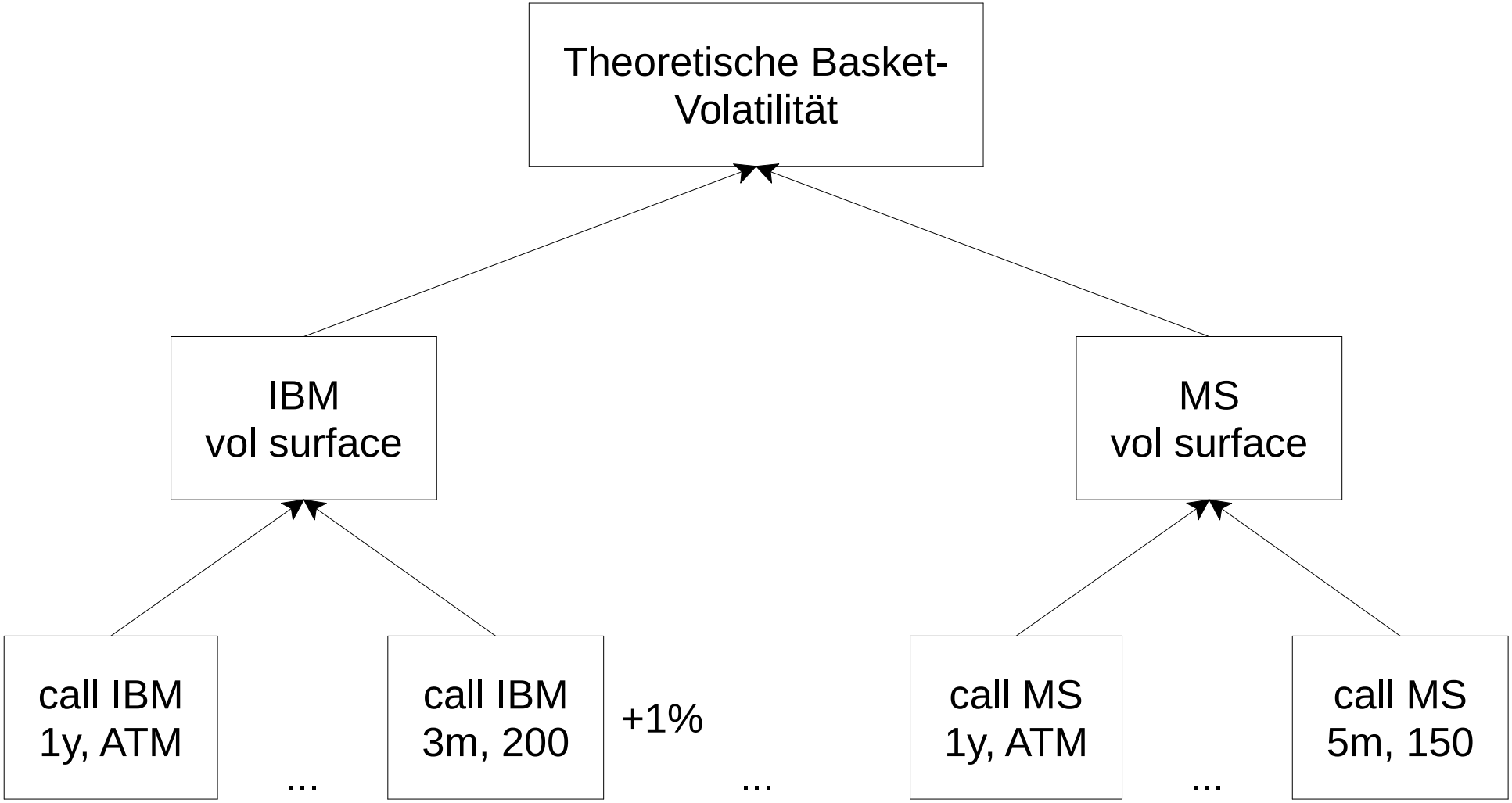
Marktdaten

- Roh:
  - Aktienkurse
  - Anleihenkurse
  - Call/Put-Kurse
- Abgeleitete Marktdaten:
  - Volatilitäten
  - Zinskurven
  - "theoretische" Marktdaten

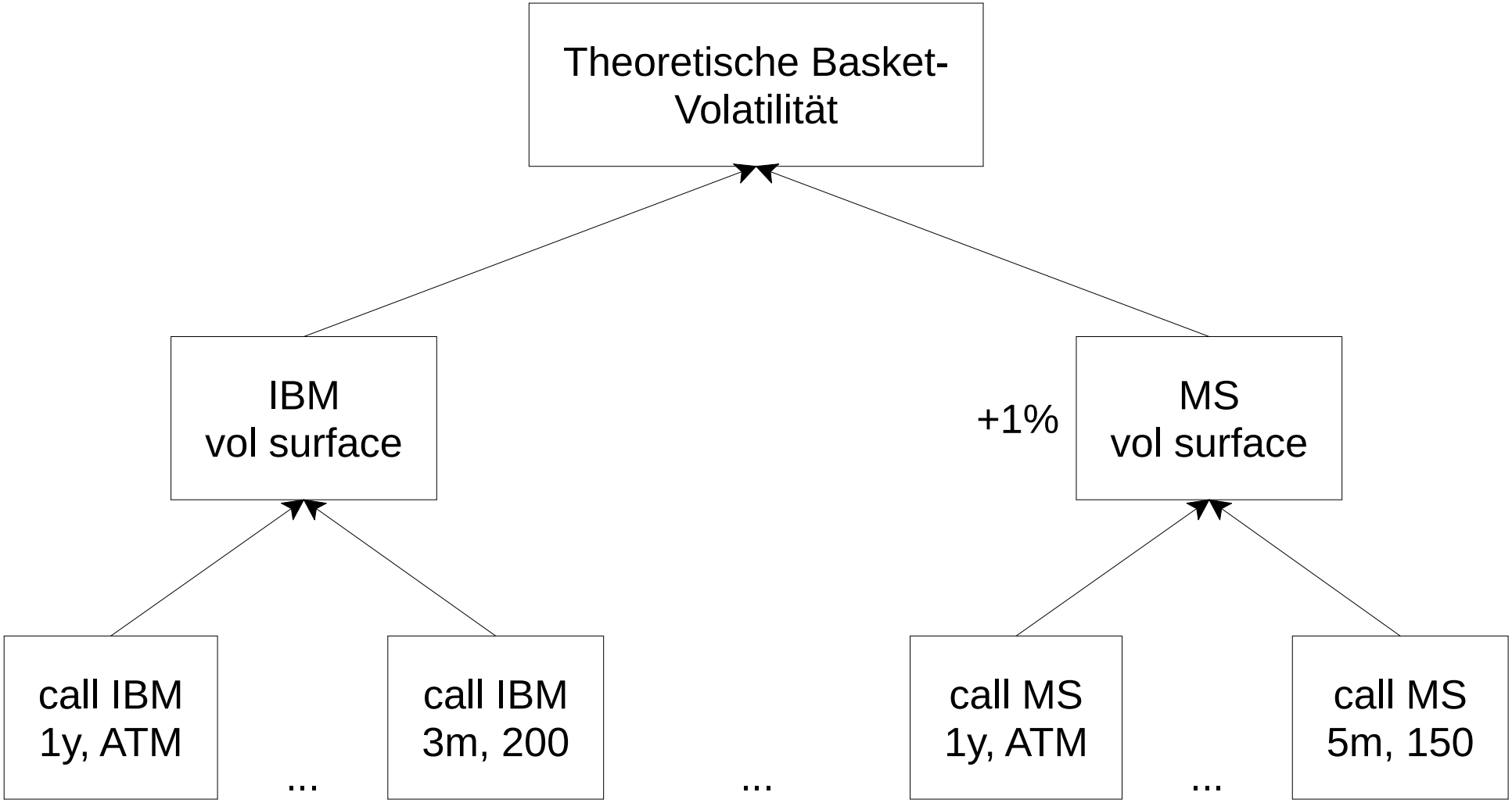
Von Kursen zu abgeleiteten Marktdaten



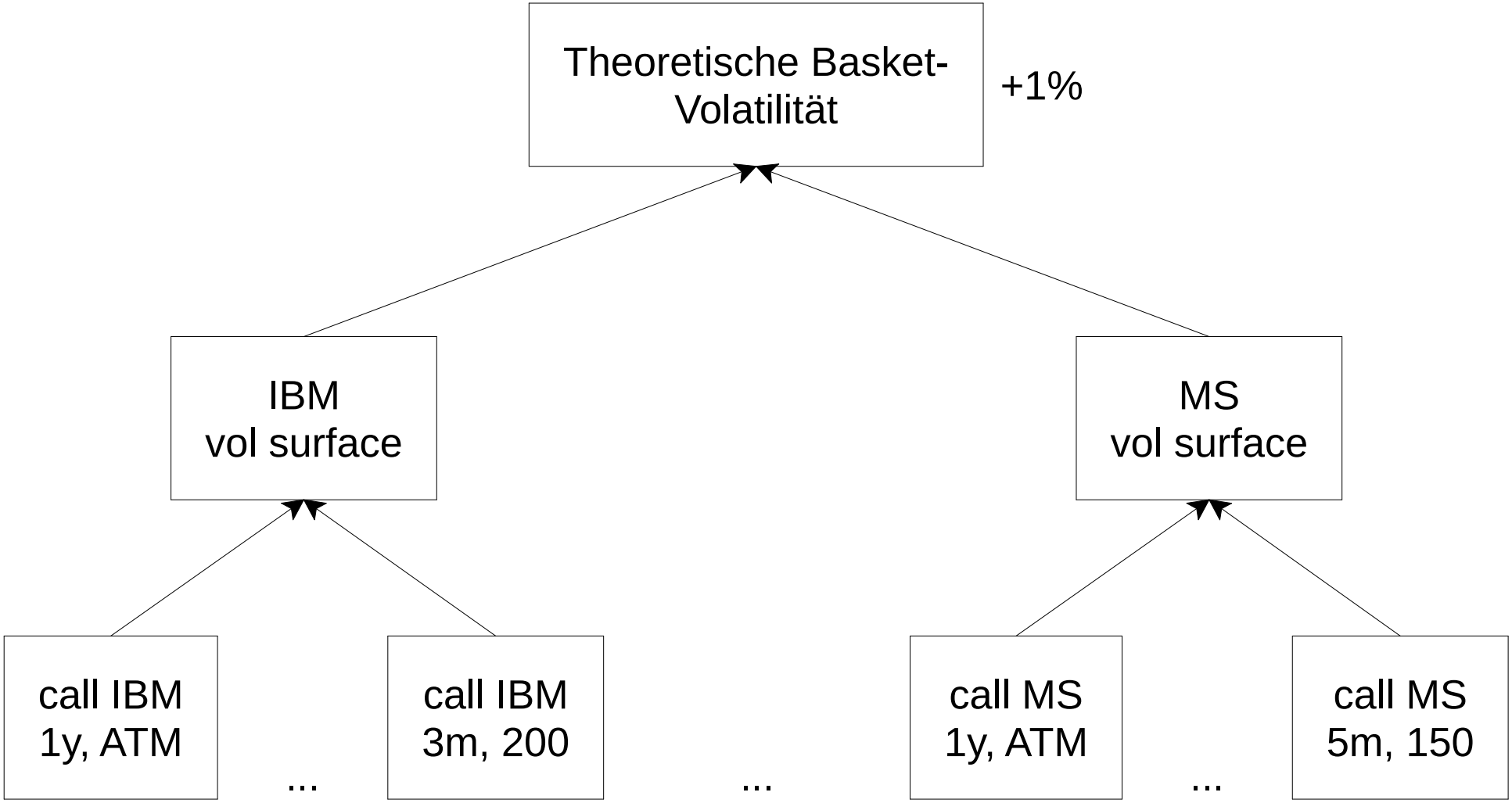
Transformationen auf Marktdaten



Transformationen auf Marktdaten



Transformationen auf Marktdaten



Klassen

- Arten von Finanzderivaten
- Quellen für Marktdaten

```
class MarketData {
   double GetSpot(long sicovam);
   double GetVolat(long sicovam,
                   double maturity, double strike);
   ...
}
```

Repräsentation transformierter Marktdaten

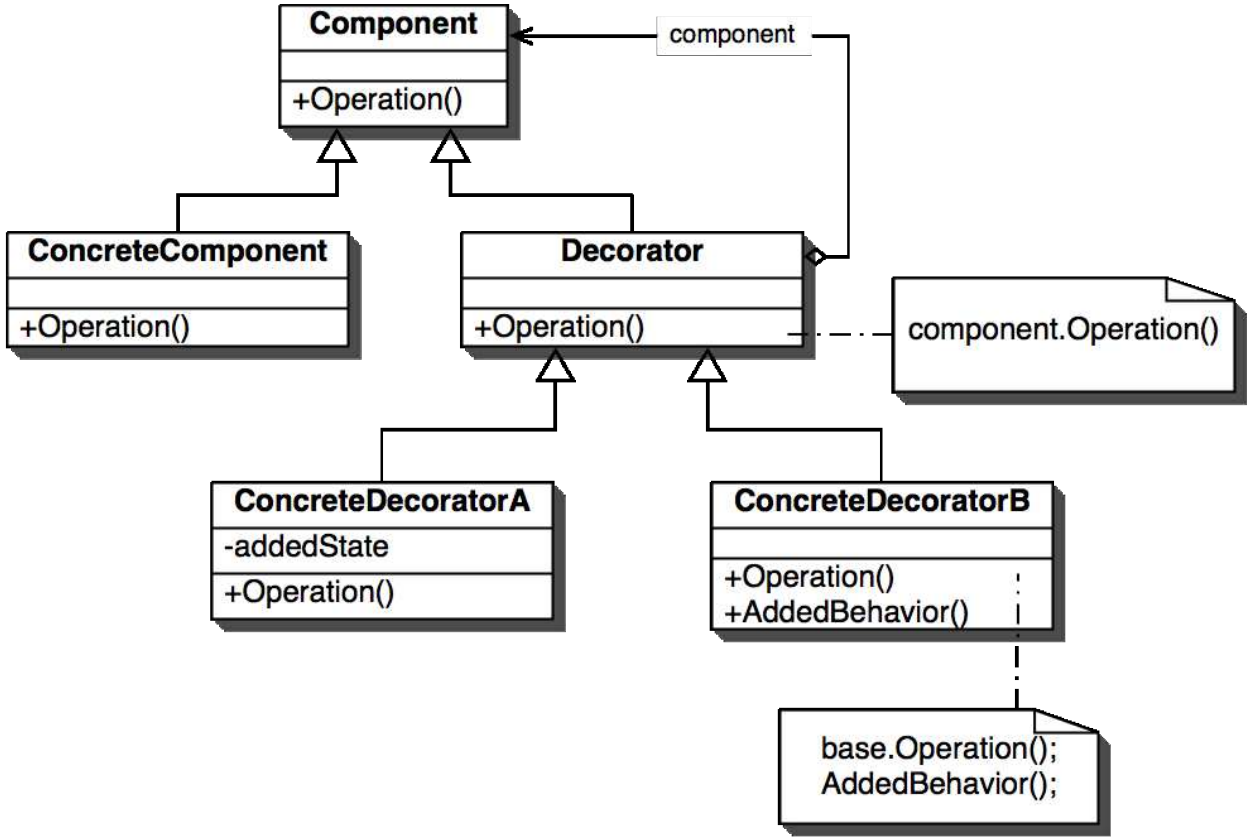
Einfach:

```
class SpotShiftedMarketData : public MarketData {
    double GetSpot(long sicovam) {
        return MarketData::GetSpot(sicovam) * factor
    }
};
```

Dynamische Komposition?

Decorator-Pattern

"Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality."



Marktdaten dekorieren ...

```
class MarketDataDecorator : public MarketData {
    MarketData* fMarketData;

    MarketDataDecorator(MarketData* marketData)
        : fMarketData(marketData) {}

    double GetSpot(long sicovam) {
        return fMarketData->GetSpot(sicovam);
    }
    // ...
};
```

Uniforme Kursverschiebung

```
double SpotShiftedMarketData::GetSpot(long sicovam)
{
  return fMarketData->GetSpot(sicovam) * factor;
}
```

Leider ...

```
double MarketData::GetSpot(long sicovam)
{
    return f(GetSpot(basketComponent), ...)
}
```

⇒ Doppelte Verschiebung!

Das Problem "beheben"

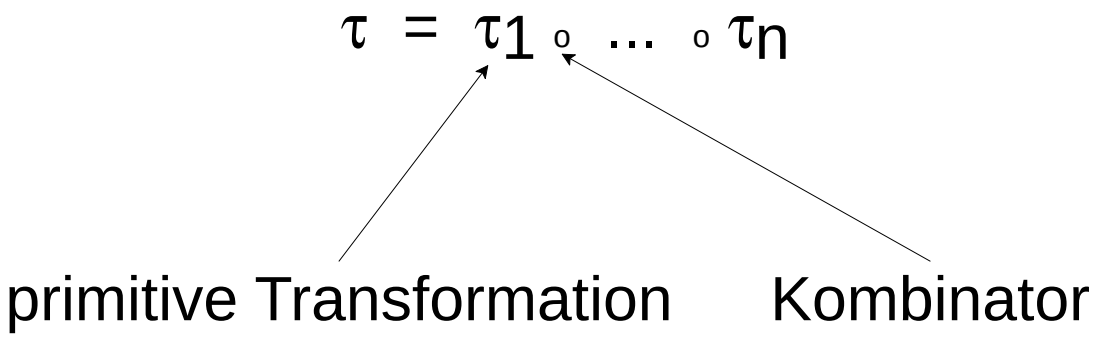
"As some method need to call other methods and so the original one instead of the overloaded one, a simple decorator is not enough; A member **`::fParameter`** has been added, every methods of **`CSRMarketData`** call the methods of **`fParameter`** instead of this and almost all methods of **`CSRMarketDataOverloader`** switch the **`fParameter`** of the overloaded market data to this; so when overloading a method, to get the original value, you must call the inherited method instead of calling directly the method of the overloaded context unless you want to avoid the method to call you back."

Leben im Alptraum

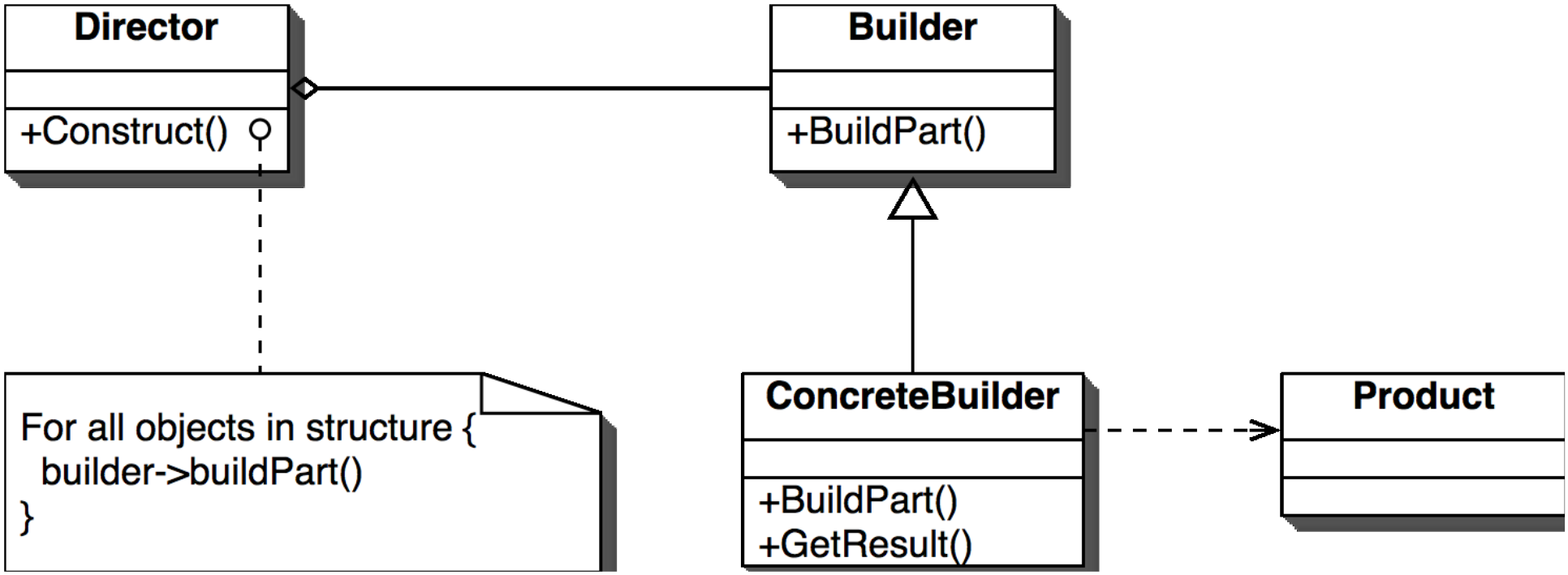
```
double  CURiskMatrixMarketData::GetVolat(   long                        code,
                                            double                      startDate,
                                            double                      endDate,
                                            double                      strike,
                                            NSREnums::eVolatilityType   volat,
                                            Boolean                     put,
                                            const CSRMarketData          *context) const
{
    if (CalledFromAPricingFunctionFromLV) {
        double init_vol = 0.0;
        const CSRInstrument *instrument = GetCSRInstrument(code);
        if(instrument
            && instrument->HasAVolatilityFormula()
            && !DoesTheFirstMarketDataDeriveFromTheSecondOne(this,context))
            init_vol = fMarketData->GetVolat(code,startDate,endDate,strike,volat,put,fMarketData);
        else
            init_vol = fMarketData->GetVolat(code,startDate,endDate,strike,volat,put,(context)?context:t

        return fabsolute_volat_shift_factor + init_vol;
    }
    else {
        const CSRInstrument *instrument = GetCSRInstrument(code);
        double vol = HVBMarketDataDelegator::GetVolat(code,startDate,endDate,strike,volat,put,context);

        if(instrument && instrument->HasAVolatilityFormula())
            return vol;
        return vol + fabsolute_volat_shift_factor;
    }
}
```

Preisfrage: Korrekt?

Modellierung von Marktdaten



Das Builder-Pattern



Dokumenten-Management-System

```
DImaObject* DImaObjectManager::GetNewImaObject(DData* pcoData)
{
  DImaObject* pcoImaObject = NULL;

  if(pcoData->GetObjectTypU().equalsLatin1("DDocument"))
  {   pcoImaObject = new DDocument;}
  else if(pcoData->GetObjectTypU().equalsLatin1("DRecherche"))
  {   pcoImaObject = new DRecherche;}
  ...
}
```

Dokumenten-Management-System

```
void DRecherche::Expand(DData* pcoDestData, DBool bRecursive, DBool bCrossOver)
{
  if(!bRecursive)
    {
      //m_pcoData->CleanDataContainer();

      if(DIMAServerModel::iFulltext)
        {
          ((DFTRechercheBuilder*)m_pcoImaObjectBuilder)
            ->VolltextRecherche(m_pcoData, pcoDestData);
        }
      else
        {
          ((DRechercheBuilder*)m_pcoImaObjectBuilder)
            ->Recherche(m_pcoData,pcoDestData);
        }

      // Flag setzen, damit asynchroner Job nach dem Zuruecksenden
      // der Objekte aktiviert wird
      m_iAsyncFlag = 1;
    } // if bRecursive
}
```

Vererbung mit Köpfchen

```
class DFTRechercheBuilder : public DRechercheBuilder
{
public:
  DFTRechercheBuilder();
  virtual ~DFTRechercheBuilder();

  void VolltextRecherche(...);
  ...
}
```

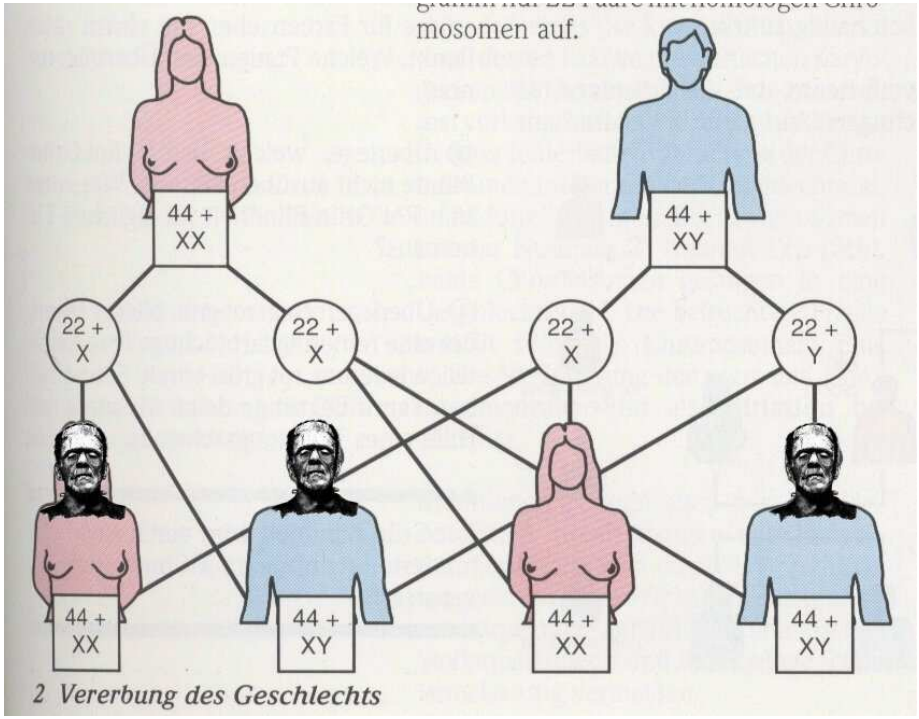
Dokumenten-Management-System

```
void DDocumentBuilder::CreateImages(DData* pcoDocumentData)
{
  ... ca. 500 Zeilen  ...

  if(!imageTitle.empty())
    pcoImageData->SetTitel(imageTitle);
  pcoImageData->ReplaceString(10, imageChecksum);
  pcoImageData->ReplaceString(11, imageFilesize);
  pcoImageData->ReplaceString(12, imageValid);
  pcoImageData->ReplaceString(13, imageTitle);

  if( DIMAServerModel::iUseCompTable )
  {
    pcoImageData->ReplaceString(33, creationDate);
    pcoImageData->ReplaceString(34, modificationDate);
    pcoImageData->ReplaceString(35, MIMEType);
    pcoImageData->ReplaceString(36, MIMECharset);
    pcoImageData->ReplaceString(37, MIMEVersion);
    pcoImageData->ReplaceString(38, compId);
    pcoImageData->ReplaceString(39, SAPStatus);
  }
  ...
}
```

Dokumenten-Management-System

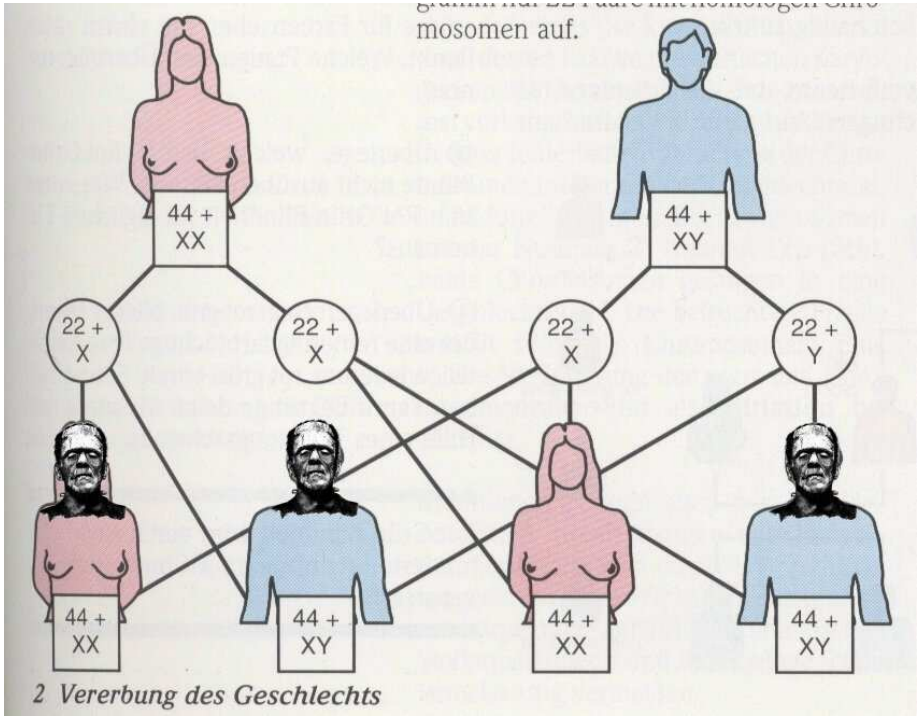
```
void DDocumentBuilde                                    ta* pcoDocumentData)
{
  ... ca. 500 Ze

  if(!imageTitle
    pcoImageData-
  pcoImageData->R                                       cksum);
  pcoImageData->R                                       esize);
  pcoImageData->Re                                      id);
  pcoImageData->Re                                      tle);

  if( DIMAServerMo
  {
    pcoImageData->                                      ionDate);
    pcoImageData->                                      ficationDate);
    pcoImageData->                                      ype);
    pcoImageData->                                      harset);
    pcoImageData->                                      ersion);
    pcoImageData->                                      d);
    pcoImageData->Re                                    tatus);
  }
  ...
}
```

Zwischenfazit



2 Vererbung des Geschlechts

Zwischenfazit



2 Vererbung des Geschlechts

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Diplomarbeit

Programmiermethodik

- CPS
- compilieren bis es durchläuft
- ausprobieren bis kein Absturz mehr
- basteln bis es "funktioniert"