„<TRADEKNOWLEDGE/>"

KONZEPTION, ENTWICKLUNG,
TEST UND BETRIEB
VON HANDELSSYSTEMEN
AUF BASIS EINES WISSENSBASIERTEN MODELLS

Detlev Matthes, dmatthes@web.de
Oktober 2022

1 INHALT

2 ABBILDUNGSVERZEICHNIS

3 TABELLENVERZEICHNIS

4 VORWORT

Die Konzeption, die Entwicklung, der Test und die reale Anwendung eines Handelssystems erwarten umfangreiche Erfahrungen und Kenntnisse in den unterschiedlichsten Bereichen. So z.B. im Verständnis von technischen Indikatoren, der Funktionsweise der Börse, den Eigenschaften des verwendeten Brokers und des zu handelnden Marktes. Weiterhin in der entsprechenden Umsetzung von Algorithmen und Handelsregeln in einer Programmierumgebung, in der Auswertung von Backtests und in der Anwendung von Handelssystemen mit „richtigem" Geld. Es müssen uns unsere Ziele und unsere Möglichkeiten bei einem Handel bewusst sein. So besitzt ein Trader mit hoher Kapitalkraft andere Möglichkeiten für einen Handel als ein Anleger mit „normalem" Einkommen. Kaum einer ist Experte in allen Gebieten und wir sind doch Individuen mit unterschiedlichen Stärken, Vorlieben und Schwächen. Jedem von uns fehlt das Wissen und die Erfahrung an irgendeiner Stelle.

Doch so viel wir auch an Zeit, Geld und Kraft investieren, um Literatur zu studieren, teure Seminare besuchen und „viel Lehrgeld" beim Traden verlieren; das gesammelte Wissen und die Erfahrungen werden lückenhaft bleiben. Kraft, Zeit und die Aufnahmekapazität unseres Gehirns sind ebenso begrenzt und zerren an unserer Gesundheit und dem privaten Umfeld. Durch unserer „menschlichen Schwächen" werden Wissen und Erfahrungen vergessen, missverstanden, falsch interpretiert und auch ignoriert. Durch Unwissenheit und falsche Interpretation von Wissen können bei der Handelssystementwicklung gravierende Fehler entstehen, die ggf. erst sehr spät oder niemals bemerkt werden und uns teuer zu stehen kommen können.

Man müsste auf das aktuelle Wissen und den Erfahrungen aller Trader, Experten und Handelssystementwickler zugreifen, dieses korrekt interpretieren und anwenden können. So könnte man sich während der Entwicklung eines Handelssystems auf das Wesentliche konzentrieren, seine Ideen umsetzen und weiterentwickeln und sich von „fremden" Ideen inspirieren lassen. Bekannte Fehler, so der unbeabsichtigte Zukunftsblick bei Backtests, können so frühestmöglich verhindert und viel Lehrgeld, Zeit und Nerven eingespart werden.

In dieser Arbeit wird eine Vorgehensweise vorgestellt, die uns dabei unterstützt, uns diesem Ziel schrittweise zu nähern. Dazu werden gesammeltes Wissen und Erfahrungen in einer Wissensdatenbank gesammelt und gespeichert. Innerhalb des Entwicklungsprozesses eines Handelssystems wird auf diese Wissensdatenbank zugegriffen und durch entsprechende Entwicklertools automatisiert interpretiert. Auf dieser Wissensdatenbank basierende Entwicklungsschritte und Tools können während des gesamten Konzeptions- und Entwicklungsprozesses, den Tests, bis hin zur praktische Betriebseinführung eines Handelssystems begleiten und unterstützten.

5 MOTIVATION UND ZIELE

Begründet durch persönliche Erfahrungen bei der Entwicklung von Handelssystemen mit deren „gemeinen" und „hinterlistigen" Fallstricken (so systematische Fehler) und dem wiederholenden Vergessen von wichtigen Aspekten (so z.B. innerhalb des Risiko- und Money-Managements) und weiterhin aus Erfahrungen von Software für die Produkt-Konfiguration ist die Idee geboren, einen auf der Produktkonfiguration basierenden Prozess für die Entwicklung von Handelssystemen zu erstellen.

Für einen Produkt-Konfigurator werden Produkte mit deren möglichen Zusammensetzungen von Produktbestandteilen mit Hilfe einer Produkt-Modellierungssprache beschrieben. Mit solch einer Modellbeschreibung ist dann die Konfiguration, also die interaktive Zusammenstellung der einzelnen Bestandteile, eines zu produzierenden Produktes möglich. Der Produkt-Konfigurator lässt dabei aufgrund der Modellbeschreibung keine inkonsistenten Zusammensetzungen zu. So kann z.B. kein Auto ohne Lenkrad produziert werden, da in der entsprechenden Modellbeschreibung definiert wurde, dass ein Auto genau ein Lenkrad besitzen muss.

Das zu entwickelnde Handelssystem wird als Produkt gesehen, dass über eine entsprechende und vorgegebene Modellbeschreibung („TradeKnowledge") interaktiv zusammengestellt, also modelliert, wird. Das somit erstellte Modell des Handelssystems ist dann entsprechend der Modellbeschreibung korrekt zusammengesetzt.

So wird z.B. bei einem LONG-System bei der Einstiegsorder ein Aktienkauf durchgeführt, da in TradeKnowledge für eine Einstiegsorder eines LONG-Systems ein Aktienkauf definiert wurde. Weiterhin werden notwendige Aspekte, wie das Risiko- und Money-Management, als verpflichtend definiert und somit bei der Modellierung als notwendige Modellbestandteile hinzugefügt. Kurz gesagt: Mit der Modellierung von Handelssystemen über einen entsprechenden Produkt-Konfigurator werden fachlich falsche Zusammensetzungen vermieden, die bei einer Programmierung mittels Programmcode (so z.B. Codierung mit Equilla von Tradesignal) möglich wären.

Stattdessen erfolgt die Erstellung des notwendigen Programmcodes automatisiert aus dem Modell des Handelssystem heraus. Mit der, für diese Arbeit entwickelte, prototypische Konfigurator-Software (im Anhang) lässt sich also Software entwickeln, die auf einer durchdachten und dem Anwendungsfall entsprechenden Architektur basiert. In unserem Fall ist es die Architektur einer Software für ein Handelssystems und dessen Bestandteile (z.B. Szenarien und Indikatoren). Aus einem einmal erstellten Modell eines Handelssystems lassen sich, entsprechende Transformatoren vorausgesetzt, Programmcodes für unterschiedliche Programmiersprachen generieren. Eine Einarbeitung in neue Programmiersprachen und die Kenntnis über dessen spezielle Eigenschaften ist dann nicht mehr notwendig.

Das mit dieser Arbeit vorgestellte TradeKnowledge dient also als Wissensspeicher über die Struktur und den Aufbau eines Handelssystems, dessen notwendige Bestandteile, die möglichen und nicht möglichen Zusammensetzungen, logischen Zusammenhänge und Interaktionen untereinander. Weiterhin unterstützt das Trade-Knowledge weitere Aspekte, so z.B. eine Unterstützung der Ideenfindung und Konzeption von Handelssystemen, der Durchführung von Tests für Bestandteile (so z.B. Indikatoren, Filter und Einstiegsetups) und der plattformunabhängigen Wiederverwendbarkeit dieser Module.

Ziele des Trade-Knowledge sind weiterhin eine Erweiterbar- und Wartbarkeit, so dass durch Trader, Experten und Entwickler stets neues Wissen und Erkenntnisse hinzugefügt und ebenso Fehler und Irrtümer korrigiert werden können.
Durch eine öffentliche Zugänglichkeit zu TradeKnowledge und den darauf basierenden Entwicklerwerkzeugen, so z.B. innerhalb einer Community, würde der Nutzen weit verbreitet und allgemein einsehbar-, nachvollzieh- und kommentierbar.

In dieser Arbeit wird ein Grundgerüst von TradeKnowledge (Kapitel 7) vorgestellt. Es soll als Ausgangbasis verstanden werden und ist für die Zukunft erweiterbar. Es werden Ideen, Konzepte und weitere interessante Anwendungsmöglichkeiten von TradeKnowledge gezeigt.

6 KNOWLEDGE UNTERSTÜTZTER ENTWICKLUNGSPROZESS FÜR EIN HANDELSSYSTEM

Im folgenden Kapitel wird der komplette Entwicklungsprozess eines Handelssystems, angefangen mit der Entwicklung der Knowledge-Sprachbeschreibung bis zum lauffähigen Programmcode innerhalb einer Handelsumgebung gezeigt.

Der Entwicklungsprozess berücksichtigt die folgenden Aspekte:

- Konzentration auf die eigenen fachlichen Interessen
    - durch die Verbergung von technischen Details der Handels- und Entwicklungsumgebung,
    - durch die Überprüfung gegen bestehendes Wissen über die Struktur von Handelssystemen,
- Umgebungsunabhängigkeit durch
    - Transformation der Handelssystem-Modelle in die zu verwendete Handelsumgebung,
    - und damit Anwendbarkeit des Handelssystems in beliebigen Handelsumgebungen
- Wiederverwendbarkeit und Anpassbarkeit von bestehenden Modellen und Bestandteile dritter (z.B. Indikatoren, Setups),
- Nachvollziehbarkeit und Robustheit durch eine einheitliche Dokumentation und verpflichtende Funktionstests,
- Einhaltung Methodik „Separation of Concerns“ (https://en.wikipedia.org/wiki/Separation_of_concerns): D.h. klare Trennung von Verantwortlichkeiten innerhalb der Prozessphasen und der Modellbestandteile

6.1 PHASEN DES ENTWICKLUNGSPROZESSES

Ein wissensbasierter Entwicklungsprozess durchläuft mehrere Phasen. Das Ergebnis einer jeden Phase ist dann ein entsprechender neuer Entwicklungszustand. Die folgende Abbildung verdeutlicht den allgemeinen wissensbasierten Entwicklungsprozess:

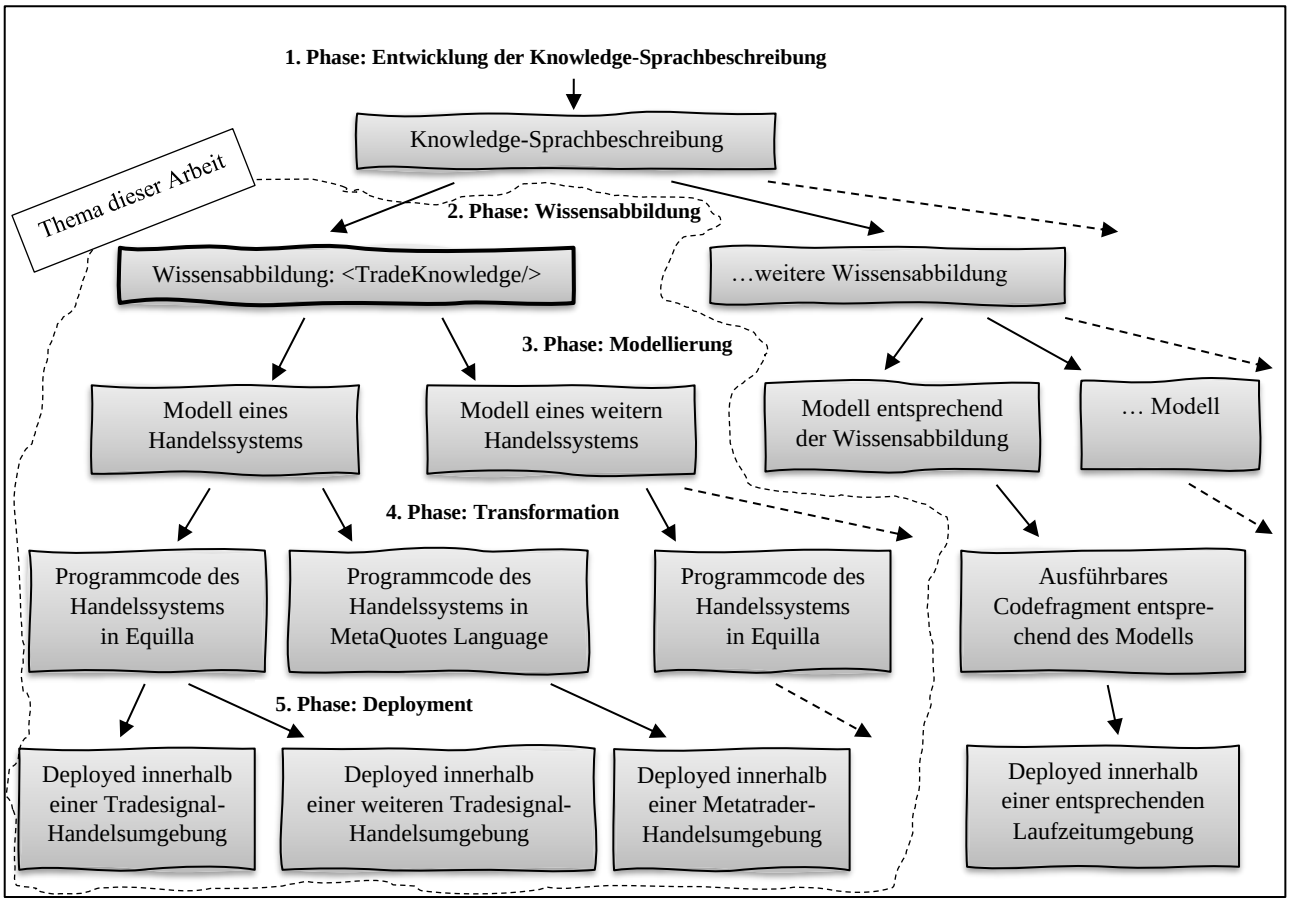

Abbildung 1: Phasen und Zustände innerhalb des Knowledge-basierten Entwicklungsprozesses

Dabei zeigt der linke eingerahmte Bereich den Entwicklungsprozess eines Handelssystems. Durch eine Verallgemeinerung der Knowledge-Sprachbeschreibung ist der Entwicklungsprozess für andere Themenbereiche adaptierbar.

Innerhalb einer Phase entstehen ein oder mehrere Zustände. So können z.B. aus einer Wissensabbildung mehrere Modelle entstehen und aus einem Modell mehrere Programmcodes für unterschiedliche Programmiersprachen transformiert werden.

Ein umgekehrter Prozess ist dabei nicht vorgesehen. So kann aus einem Programmiercode kein Modell zurücktransformiert werden und aus einem Modell keine Wissensbeschreibung erstellt werden. Solch ein Vorgehen, würde den Prozess unnötig verkomplizieren.

Erfolgen Änderungen (so z.B. Verbesserungen und Korrekturen) innerhalb eines Zustandes, so können die darunterliegenden Prozessphasen wiederholt werden. So kann z.B. ein bestehendes Modell eines Handelssystems von den Verbesserungen an TradeKnowledge profitieren. Des Weiteren kann TradeKnowledge von Verbesserungen an der Wissensbeschreibung profitieren. Durch eine toolunterstützte Durchführung der Phasen, kann die wiederholte Durchführung automatisiert erfolgen.

Die Durchführung und Auswertung von Tests ist in den einzelnen Prozessphasen integriert. So erfolgen z.B. in der Deploymentphase eine Überprüfung, ob das Handelssystem innerhalb der Handelsumgebung ausgeführt werden kann und notwendige Vorrausetzungen erfüllt werden.

In den folgenden Kapiteln werden diese Phasen erläutert. Der Schwerpunkt dieser Arbeit liegt in der Wissensabbildung in TradeKnowledge (2. Phase) und wird ausführlich im Kapitel 7 behandelt.

6.2 1. PHASE: ENTWICKLUNG DER WISSENS-SPRACHBESCHREIBUNG „KNOWLEDGE“

Um Wissen abbilden und beschreiben zu können ist eine entsprechende Sprache notwendig. In dieser Phase erfolgt die Entwicklung der Wissens-Sprachbeschreibung: „Knowledge“.

Die Basis bilden Entitäten. Diese stellen Objekte in der realen Welt dar. Beispiele für Entitäten sind ein Handelssystems, ein Indikator oder ein Einstiegssetup. Entitäten und deren Eigenschaften und Beziehungen werden durch Klassen beschrieben. Eine Notation kann mittels der Unified Modelling Language (https://www.uml.org/) erfolgen.

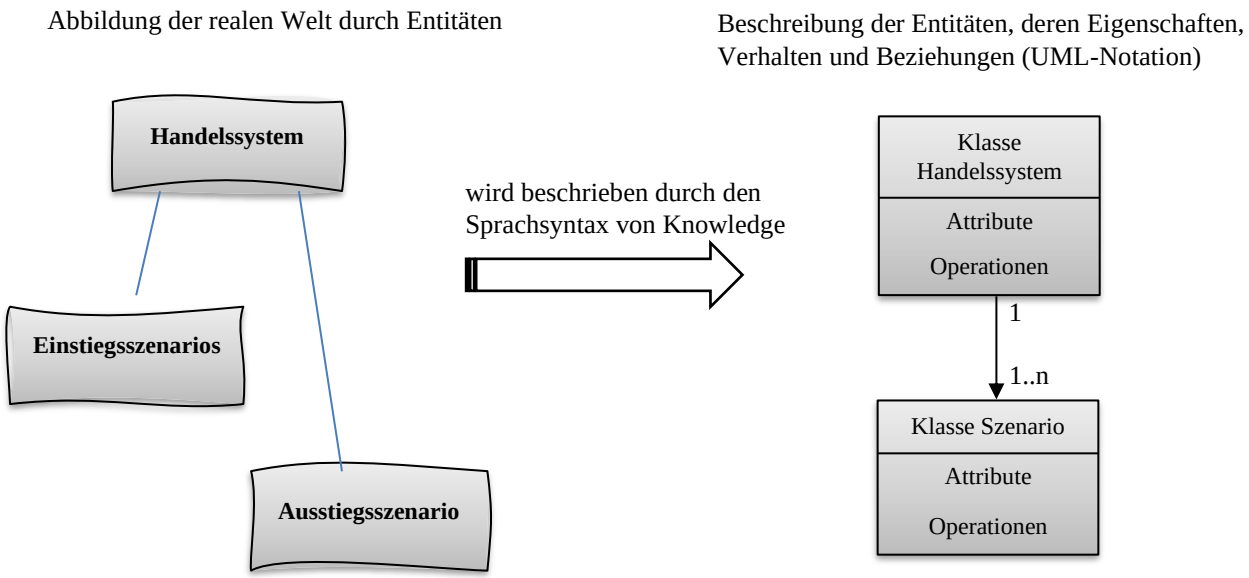

Abbildung 2: Beispiel Abbildung von Entitäten und deren Beschreibungen

Mit einem entsprechenden Sprachsyntax erfolgt die Beschreibung dieser Klassen mit deren Eigenschaften (Attribute), Verhalten (Operationen) und Beziehungen (Assoziationen) untereinander. Durch eine Beschreibung dieser Eigenschaften kann dann Wissen einer Domaine (so z.B. die Struktur eines Handelssystems) abgebildet werden.

Zum weiteren Verständnis:

- Attribute von Klassen werden mit einem vorangestellten @-Zeichen (so @name) notiert.
- Eine aus einer Klasse gebildeten Entität (auch Instanz) besitzt das Standardattribut @name als eindeutigen Namen der Entität. So entspricht der Name eines modellierten Handelssystems dem Wert von @name.

6.3 2. PHASE: WISSENSABBILDUNG IN „TRADEKNOWLEDGE“

Innerhalb der Wissensabbildung werden das Wissen und die Erfahrungen über die Entwicklung von Handelssystemen als „TradeKnowledge“ in der Syntax von „Knowledge“ abgebildet.

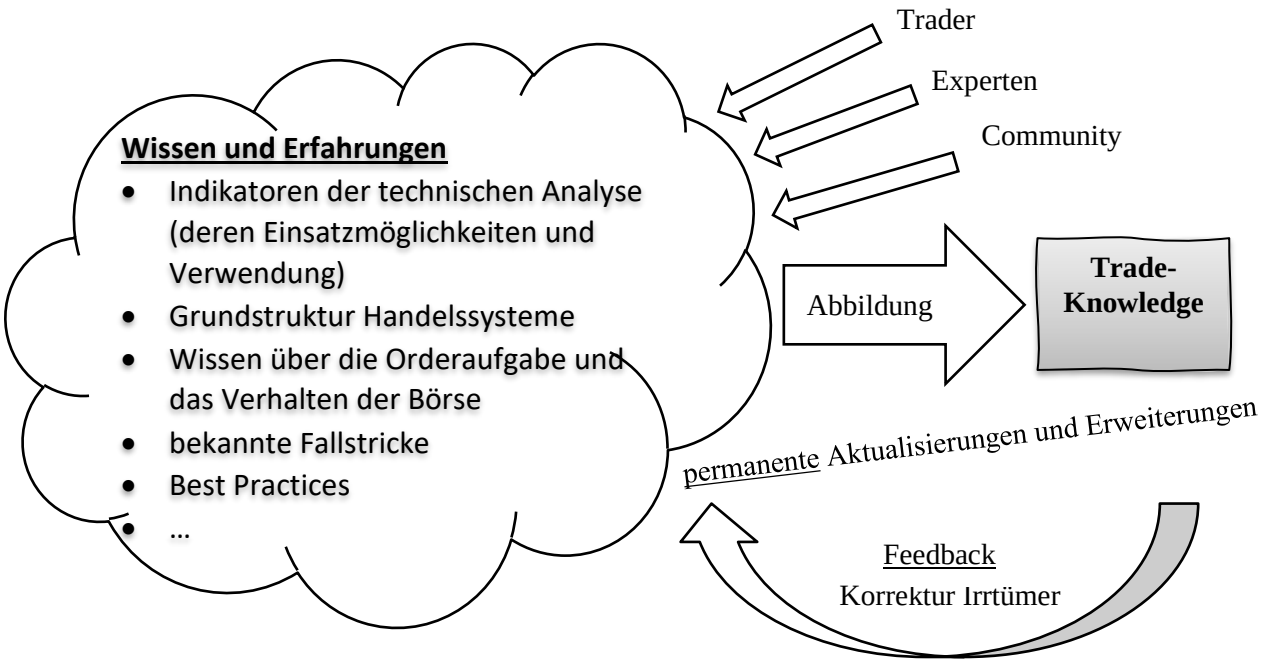

Abbildung 3: Abbildung von Tradingwissen in TradeKnowledge

Die Wissensbeschreibung ist ein stätiger und interaktiver Prozess. TradeKnowledge wird also fortlaufend um neues Wissen erweitert und korrigiert. Bereits erstellte Modelle von Handelssystemen, mit denen ggf. schon real gehandelt wird, können gegenüber eines aktualisierten TradeKnowledge erneut validiert werden und profitieren somit vom neuen Wissen und Korrekturen.

6.4 3. PHASE: MODELLIERUNG

Innerhalb der Modellierungsphase werden aus Idee, Skizzen und Konzepten ein Modell des gewünschten Handelssystems erstellt.

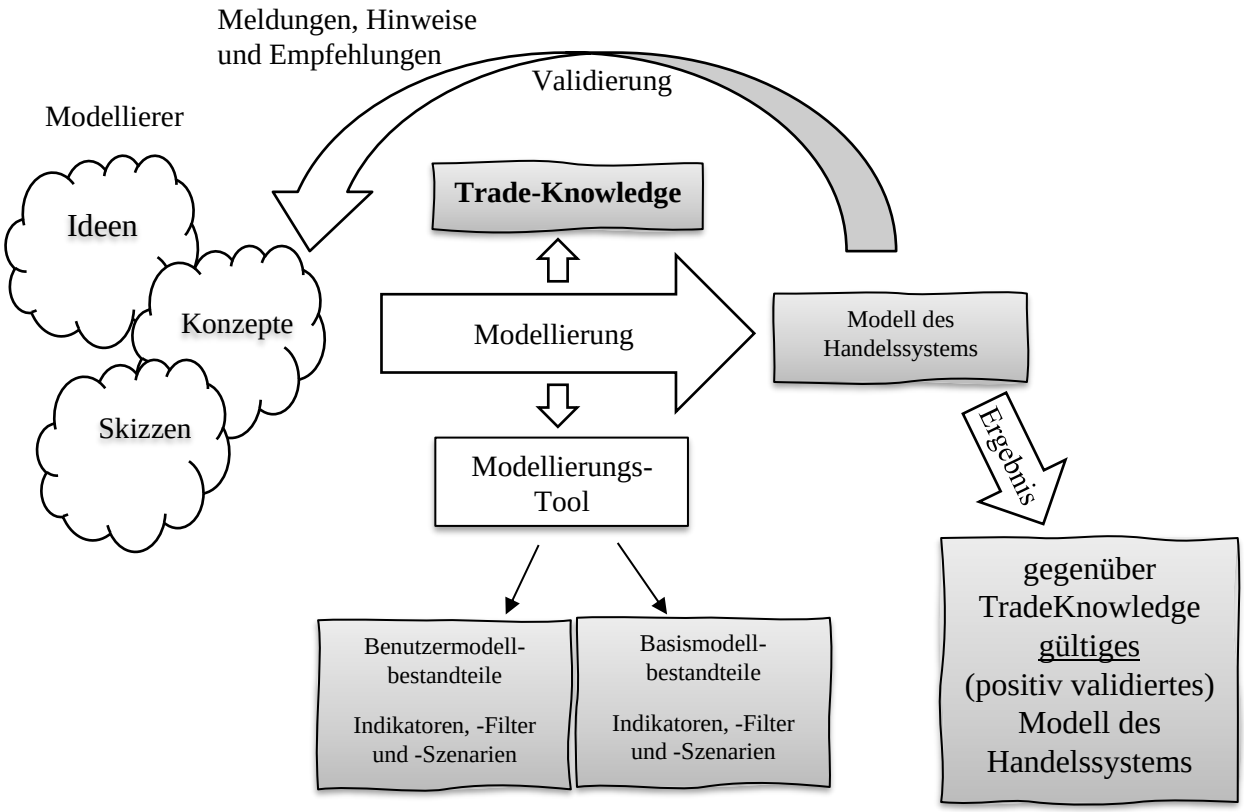

Abbildung 4: Modellierung eines Handelssystem mit gleichzeitiger Validierung gegen TradeKnowledge

Entsprechend der in TradeKnowledge abgebildeten Wissensbeschreibung erstellt der Modellierer mithilfe eines Modellierungs-Tools das Modell des gewünschten Handelssystems. Verpflichtende Komponenten werden als Modellbestandteile hinzugefügt. Alle Komponenten, wie z.B. Szenarien, Indikatoren, Filter u.a. können nur entsprechend Ihrer Beschreibung zusammengefügt werden. Unerlaubte Zusammensetzungen und Konstellationen entsprechen einem nicht-gültigen Modell. Während der Validierung des Modells gegen TradeKnowledge erzeugte Hinweise (z.B. für Verbesserungsvorschläge) und Fehlermeldungen (für z.B. für falsche Zusammensetzungen) werden vom Modellierer verwendet, um entsprechende Anpassungen oder Verbesserungen am Modell vorzunehmen (siehe 7.3.1).

Die Modellierung ist also ein interaktiver Prozess von einer wiederholenden Modellierung und Validierung bis ein gültiges, d.h. ein gegen TradeKnowledge positiv validiertes Modell entstanden ist, das schließlich die Ideen, Skizzen und Konzepte des Handelssystems enthält.

Nur solche, gegenüber TradeKnowledge, gültige Modelle können in den weiteren Entwicklungsphasen weiter prozessiert werden.

Innerhalb einer Community können Entwickler Indikatoren, Filter und Szenarien als fertig modellierte Bestandteile zur freien Verfügung stellen. Diese können dann eingebunden, verbessert und erweitert werden.

6.5 4. PHASE: TRANSFORMATION

In der Transformationsphase werden mithilfe eines Transformator-Tools aus dem Modell des Handelssystems der Programmcode des Handelssystems (ggf. auch Konfigurationsdateien und Installationsskripte/Anweisungen) entsprechend der gewünschten Zielumgebung erzeugt. So wird z.B. mit Transformatoranweisungen für die Zielumgebung AFL (für AmiBroker) entsprechender Programmcode in der Programmiersprache AFL aus

einem Modell heraus generiert. Aus demselben Modell kann ebenfalls mit Transformatoranweisungen für die Zielumgebung ProRealTime ein Programmcode in der Programmiersprache ProBuilder generiert werden.

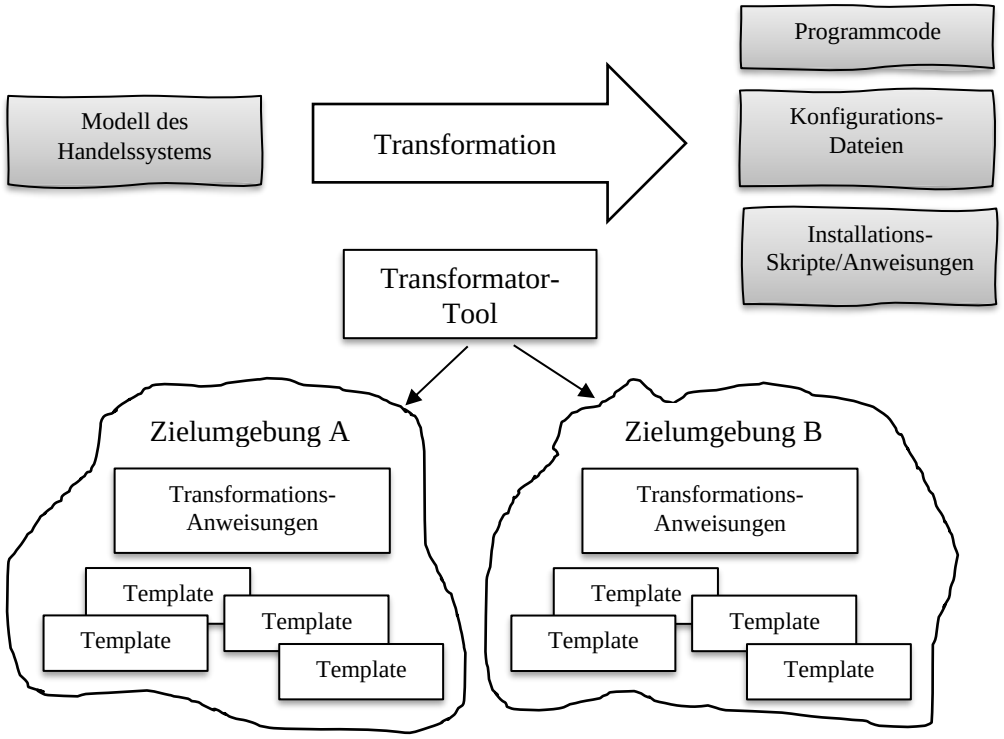

Abbildung 5: Transformation von ausführbaren Programmcode aus dem *Modell heraus*

Entsprechend der Zielumgebung werden durch das Transformator-Tool entsprechende Transformationsanweisungen durchgeführt. Diese verwenden Templates in der Programmiersprache der Zielumgebung (so z.B. ein Template zur Berechnung eines gleitenden Durchschnitts).

Mithilfe der Transformation ergeben sich gegenüber der klassischen Programmierung von Handelssystemen die folgenden Vorteile:

- keine Einarbeitung in die Programmiersprache und Entwicklungsumgebung notwendig
- Best-Praxis-Ansätze und „geheime“ Tricks in der Entwicklung müssen nicht mühsam erarbeitet werden, sondern „stecken“ bereits innerhalb der Transformationsanweisungen

Desweitere wird durch die Transformation spezielles Verhalten und Besonderheiten der Handelsumgebung, so z.B. mit der Überwachung des Portfoliokapitals, in der Darstellung von Indikatoren, in der Übergabe von zu optimierenden Parametern und vieles mehr, gekapselt.

Neben den Algorithmen des Handelssystems werden Programmcodes für weitere notwendig Aspekte zusätzlich generiert:

- Prüfmethoden zur Generierung von Meldungen zur Laufzeit (siehe 7.3.2)
- Visualisierungen von Indikatoren, Handelssignalen und Orderanweisungen im Chart
- Schalter, die das Verhalten des Handelssystem steuern. So z.B. Schalter für die Aktivierung/Deaktivierung eines Trendfilters
- Methoden zum Monitoring und Überwachen während der Laufzeit (so z.B. Vergleich mit realen Trades und Ermittlung von Kennzahlen wie z.B. max. Drawdown und Profitfaktor)

6.6 5. PHASE: DEPLOYMENT

In dieser letzten Phase werden der erstellte Programmcode und andere notwendige Bestandteile in die Zielumgebung installiert.

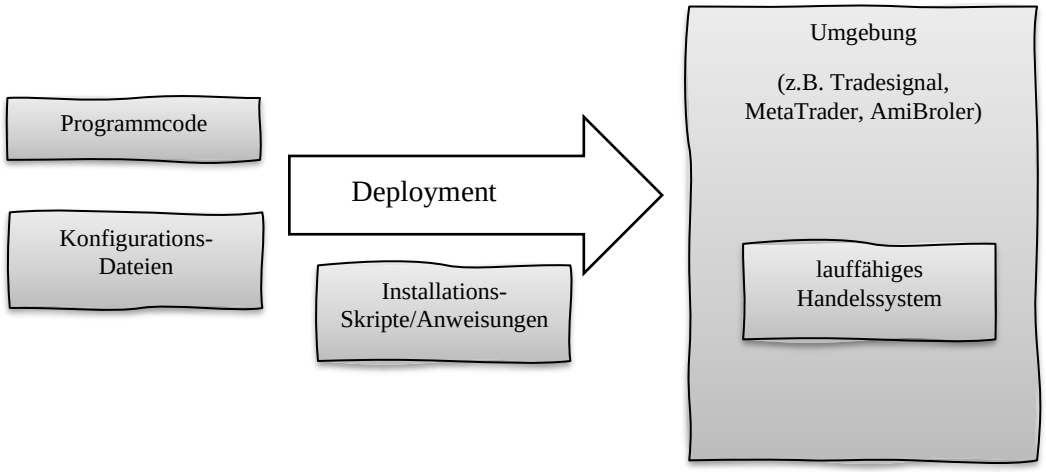

Abbildung 6: Deployment des Handelssystems in die Zielumgebung

Dieser Schritt erfolgt manuell oder automatisiert durch entsprechende Installationsskripte (die entsprechend umgebungsabhängig sind).

Während und nach dem Deployments kann überprüft werden, ob die Handelsumgebung die Anforderungen an das Handelssystem erfüllt, z.B. ob die Kursdaten in den geforderten Zeitintervallen zur Verfügung stehen.

Somit ist das entwickelte Handelssystem in der gewünschten Handelsumgebung betriebsfähig.

7 TRADEKNOWLEDGE

Das folgende Kapitel behandelt die Abbildung des Wissens über die Struktur, Funktionsweise von Handelssystemen und Best-Praxis-Ansätzen in TradeKnowledge. Weiterhin werden gängige Fehler in der Handelssystementwicklung erwähnt und deren Vermeidung durch TradeKnowledge beschrieben.

7.1 GRUNDLEGENDE STRUKTUR EINES HANDELSSYSTEMS

Im Folgenden wird die grundlegende Struktur eines, in TradeKnowledge modellierbaren Handelssystems beschrieben.

Um die Komplexität in Grenzen zu halten, fokussiert sich die hiermit vorgestellte Version von TradeKnowledge auf einen einfachen Aktienhandel mit LONG-Positionen und ohne den teilweisen Auf- und Abbau (Pyramidisierung) von Positionen auf ein Portfolio. Angewendet werden soll das Handelssystem von Berufstätigen (ohne Tradingbezug).

Ein Handelssystem besteht aus Einstiegs- und Ausstiegszenarien. In einem Einstiegsszenario wird eine LONG-Aktienposition eröffnet und in einem Ausstiegsszenario eine LONG-Aktienposition komplett geschlossen. Weiterhin wird ein einfaches Risiko- und Money-Management unterstützt.

In den folgenden Kapiteln werden die grundlegenden Bestandteile (Klassen, Attribute und deren Beziehungen untereinander) erwähnt. Für die Darstellung wird die Unified Modelling Language (UML) verwendet.

7.1.1 Modellklasse: Handelssystem

Ein Handelssystem wird durch eine Instanz der Klasse „Tradingsystem“ abgebildet.

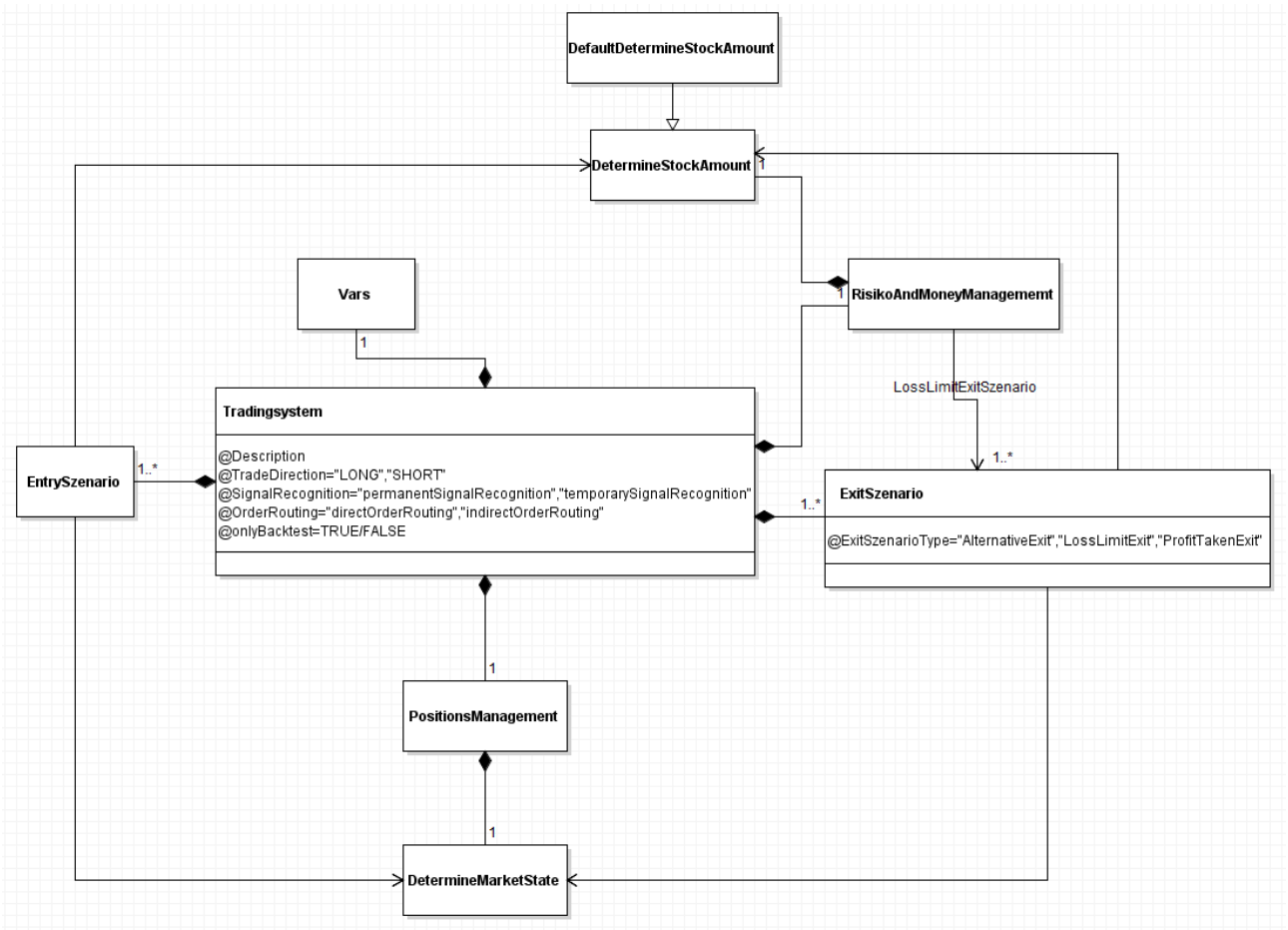

Abbildung 7: Modellstruktur Handelssystem (engl. Tradingsystem)

Ein **Tradingsystem** besteht aus mehreren Szenarien (siehe 7.1.2). Davon muss mindestens ein **EntrySzenario** und mindestens ein **ExitSzenario** vorhanden sein.

Über die folgenden Attribute der Klasse Tradingsystem werden grundlegende Eigenschaften des Handelssystems beschrieben:

- @name bzw. der Name der entsprechenden Klassenausprägung (Instanz) betitelt das Handelssystem. *Hinweis: jede Klasse besitzt das Attribut @name.*
- @Description betitelt und beschreibt textuell ein Handelssystem und dient so einer klaren Dokumentation.
- @SignalRecognition gibt an, mit welcher Laufzeit das Handelssystem neue Handelssignale ermittelt. So „permanentSignalRecognition“ für eine selbstständige Signalermittlung, bei dem das Handelssystem während der gesamten Handelszeit aktiv ist und so zeitnah mit jeden neuen Datenfeed entsprechende Handelssignale generieren kann. Oder „temporarySignalRecognition“, wenn das Handelssystem nur zum Zwecke einer Signalermittlung ausgeführt wird, so z.B. ein Berufstätiger überprüft nach Feierabend das Handelssystem nach neuen Kauf- oder Verkaufssignalen.
- @OrderRouting gibt an, ob das Handelssystem Orders direkt an den Broker gibt (z.B. über eine API) oder ob die Order indirekt, z.B. manuell über eine Eingabemaske (z.B. innerhalb einer Webseite), an den Broker übergeben werden.
- @onlyBacktest gibt an, ob das Handelssystem zu Experimentier- und Forschungszwecken nur als Backtest entwickelt werden darf und somit ein (ggf. unbeabsichtigter) realer Einsatz ausgeschlossen ist.

RisikoAndMoneyManagement für das Risiko- und Moneymanagement ist Pflichtbestandteil eines Handelssystems und muss einen Verweis auf mindestens ein ExitSzenario für eine Risikobegrenzung (ExitSzenario-Typ=“LossLimitExit“) enthalten. Somit besitzt ein Handelssystem mindestens ein Exitszenario für eine Risikobegrenzung, ebenfalls als ein Pflichtbestandteil. Mit dem Attribut „InitialStopPriceAtEntry“ wird der Stopp-Preis für eine Verlustbegrenzung angegeben. Das kann z.B. der prozentuale Abstand zum Einstiegspreis oder der Wert eines Indikators zum Zeitpunkt des Einstiegs sein.

Mit **DetermineStockAmount** enthält das Risiko- und Moneymanagement eine Funktionalität zur Bestimmung von Größen für einzugehende Positionen zur Risikobegrenzung. Mit **DefaultDetermineStockAmount** wird eine Standardmethode bereitgestellt, die die Positionsgröße auf Basis 1%-Risiko des Portfoliokapitels bestimmt. Alternativ können andere bereitgestellte oder eigene Strategien verwendet werden.

PositionsManagement ist ebenso ein Pflichtbestandteil eines Handelssystems. Mit **DetermineMarketState** erhält das Positionsmanagement eine Funktionalität um den Status vom Handelssystem erstellten Positionen zu ermittelt. Also z.B. wieviel AAPL-Aktien (Apple-Aktie) wurden insgesamt gekauft und befinden sich aktuell im Portfolio.

In einem **Vars**-Bestandteil werden die Werte von, im Handelssystem verwendeten Indikatoren, Filtern, Konstanten und Eingabeparameter, gehalten. Der Vars-Bestandteil wird im Anhang zu dieser Arbeit detaillierter erläutert.

Hinweis: Filter (so z.B. ein Trendfilter für die Filterung von Einstiegsszenarien in trendstarken Phasen) werden z.Zt. noch nicht von TradeKnowledge unterstützt.

7.1.2 Modellklasse: Szenario

Innerhalb eines **Szenario**s werden auf Basis eines Algorithmus Handelssignale und entsprechende Orders für einen Handel generiert. Es wird z.Zt. unterschieden nach Szenarien für den Einstieg (EntrySzenario) in eine Position und den Ausstieg (ExitSzenario) aus einer Position.

Beispiel für Szenarien:

- Einstiegsszenario: Gehe Long, wenn ein gleitender Durchschnitt einen anderen gleitenden Durchschnitt von unten nach oben kreuzt
- Ausstiegsszenario: Schließe Long-Position über eine bestehende Stopp-Order zum Stopp-Kurs für die Verlustbegrenzung

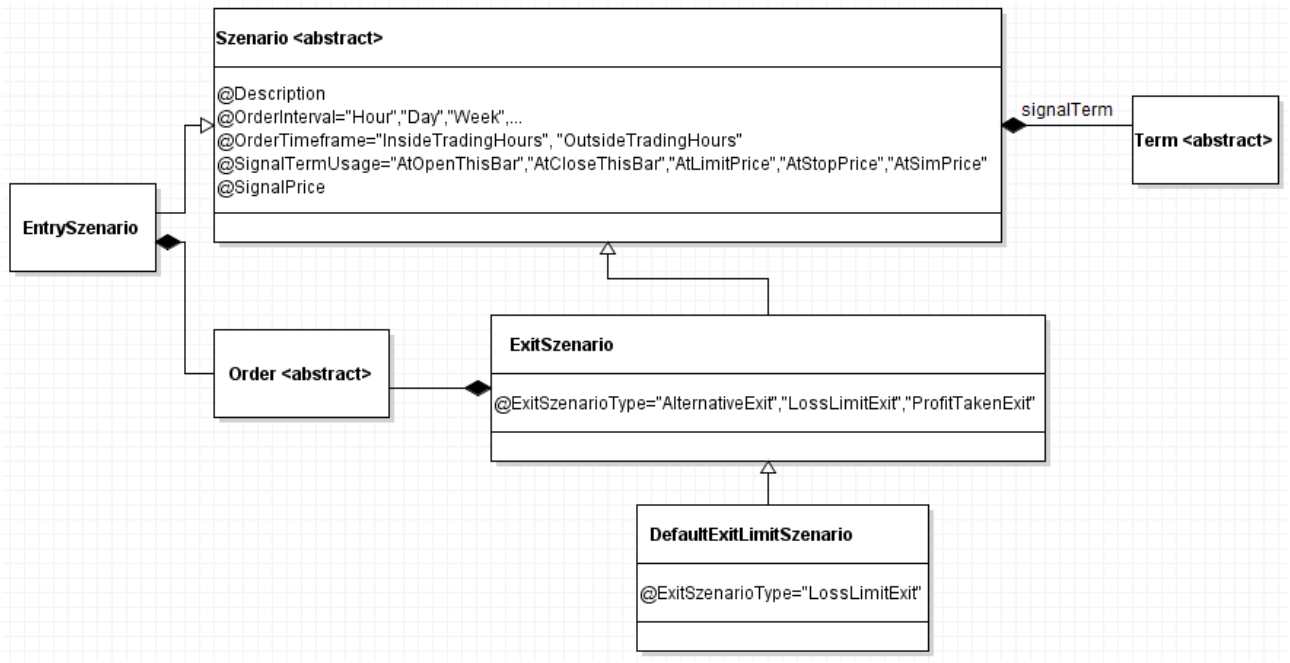

Abbildung 8: Modellstruktur Szenario

Mit @Description wird das Szenario textuell beschrieben und dient so einer klaren Dokumentation.

Ein Szenario enthält einen booleschen **Term** (signalTerm) mit dem der Algorithmus zur Generierung eines Handelssignals abgebildet wird. Liefert der Term TRUE, so ist das Handelssignal gegeben. Terme werden im Anhang zu dieser Arbeit detailliert erläutert. Für ein Einstiegsszenario mit einer einfachen Kreuzung von zwei gleitenden Durchschnitten kann der vordefinierte **CrossOver**-Term (siehe Anhang, Kapitel 6.2.2) verwendet werden.

Innerhalb eines Ausstiegszenarios (ExitSzenario) wird mit @ExitSzenarioType der Ausstiegstyp angegeben. So wird mit LossLimitExit das Ausstiegszenarios für eine Verlustbegrenzung gekennzeichnet.

Ein SignalTerm kann ebenfalls aus einer Kombination von Termen bestehen (siehe Anhang, Kapitel 6.2.3). So z.B. eine logische UND-Verknüpfung eines CrossOver-Terms und eines Terms für die Erkennung einer „Outside-Bar“-Candlestick-Formation.

7.1.2.1 Gültigkeit von Handelssignalen

Entsprechend dem, innerhalb des SignalTerms, verwendeten Werten ergeben sich für den SignalTerm Gültigkeiten. So ist ein Term für ein Handelssignal, der den Close-Preis des aktuellen Tages und den Low-Preis der aktuellen Woche verwendet, erst mit Abschluss der aktuellen Woche gültig. Zusammen mit dem Attributwert von @OrderInterval erfolgt eine Überprüfung der Gültigkeit des SignalTerms durch TradeKnowledge.

Über das Attribut @OrderInterval wird das beabsichtigte Intervall für einen Handel festgelegt. Mögliche Werte wären u.a. „Day“ oder „Week“. So sollen z.B. mit „Week“ wöchentlich gültige Handelssignale generiert werden.

7.1.2.1.1 Gültigkeit von Preisen innerhalb eines Intervalls

Die folgende Abbildung zeigt beispielhaft den Ausschnitt aus zwei Wochen-, der entsprechenden Tagesintervalle und den Tick-Preisen.

Hinweis: TradeKnowledge betrachtet noch nicht das Volumen

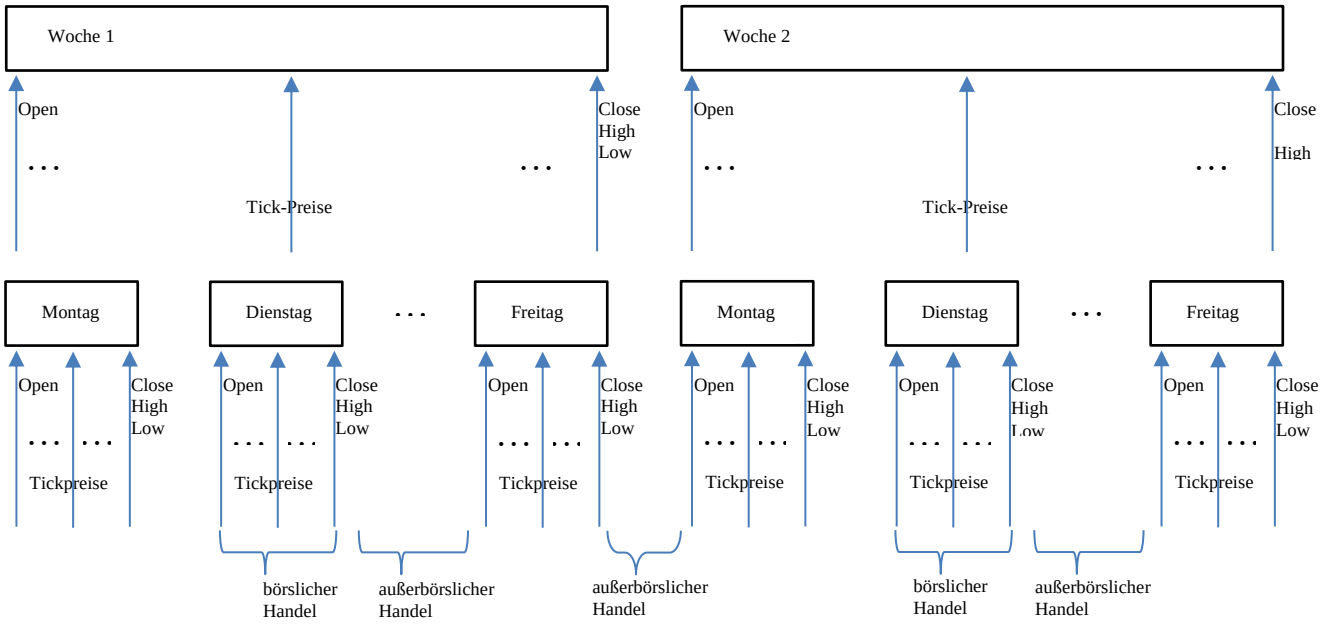

Abbildung 9: Gültigkeit von Preisen

Es ist zu erkennen:

- Die Werte von Open, High, Low und Close (kurz OHLC) ergeben sich aus den Tick-Preisen.
- Der Open-Preis entspricht dem ersten Tick-Preis eines Intervalls.
  Der Eröffnungskurs ist also mit Beginn eines Intervalls gültig. Der Open-Preis kann somit innerhalb eines Intervalls zur Signalgenerierung in einem Term und für einer Orderaufgabe verwendet werden.
- Der Close-Preis entspricht dem letzten Tick-Preis eines Intervalls. Innerhalb eines noch nicht abgeschlossenen Intervalls ändert sich der Close-Preis entsprechend dem aktuellen Tick-Preis.
  Der Close-Preis ist also erst nach Abschluss des Intervalls gültig und kann somit erst nach Abschluss des Intervalls zur Signalgenerierung in einem Term und für einer Orderaufgabe verwendet werden.
- Der Low- und High-Preis entspricht dem minimalen bzw. maximalen Tick-Preis eines abgeschlossenen Intervalls. Innerhalb eines noch nicht abgeschlossenen Intervalls können stets neuen Minimums und Maximums entstehen.
  Low- und High- Preise sind also erst nach Abschluss des Intervalls gültig und können somit erst nach Abschluss des Intervalls zur Signalgenerierung in einem Term und für einer Orderaufgabe verwendet werden.

In der Praxis wird häufig der Fehler gemacht den Abschluss eines Intervalls nicht abzuwarten. Die dann noch nicht gültigen LHC-Preise (Low, High und Close) werden bereits zur Auswertung eines Algorithmus zur Signalgenerierung verwendet. Dadurch können Fehlsignale (ugs. Flattersignale) entstehen.

Werden die LHC-Preise des aktuellen Intervalls innerhalb des Signalterms verwendet, muss also eine Signalgenerierung und eine damit verbundene Ordergenerierung innerhalb des aktuellen Intervalls unterbunden werden. Nur Open-Preise des aktuellen Intervalls und LHC-Preise aus abgeschlossenen Intervallen dürfen somit innerhalb des Signalterms verwendet werden.

Zur Vermeidung dieser Fehler, werden die folgenden Regeln definiert und von TradeKnowledge ausgewertet:

- Verwendung des, innerhalb eines Szenarios festgelegten, Intervalls (Szenario@OrderIntervall) mit der eine Auswertung des Signalterms erfolgt (so z.B. „Day“ oder „Week“).
- Werden innerhalb eines Signalterms aktuelle LHC-Preise des gleichen Intervalls wie Szenario@OrderIntervall verwendet, so wird eine Auswertung des Signalterms und eine damit verbundene Ordergenerierung innerhalb des Intervalls unterbunden. Dieses erfolgt durch Anwendung von zeitlichen Filtern.
  So wird z.B. bei Szenario@OrderIntervall=Day und bei einem Handel auf XETRA die Auswertung von Signaltermen von 9:00Uhr bis 17:30Uhr unterbunden. Erst nach 17:30 stehen die LHC-Preise fest und die Auswertung von Signaltermen kann erfolgen.
  In einer der nächsten Versionen von TradeKnowledge, könnten innerhalb der Zeiten, in denen eine Unterbindung stattfindet, Vorsignale erstellt werden die dann zur Vorbereitung von möglichen Trades oder einem ggf. vorgenommenen vorzeitigen Handel (so z.B. 15 Minuten vor Börsenschluss) verwendet werden können.

Hinweis: ein Daytrader sollte somit stets ein passendes niedriges Orderintervall (so z.B. 15min) verwenden.

7.1.2.1.2 Mischung von Preisen unterschiedlicher Intervalle

Werden in einem Signalterm LHC-Preise unterschiedlicher Intervalle miteinander kombiniert, so muss für Szenario@OrderIntervall das höchste Intervall verwendet werden. Werden also der Close-Preis des Tages (niedriges Intervall) und der Low-Preis der Woche (höheres Intervall) zusammen in einem Signalterm verwendet, so muss die Auswertung der Signale im Wochenintervall erfolgen. Wie in der Abbildung 9 zu erkennen ist, ist zwar am Dienstag der Close-Preis von Montag gültig, aber der Low-Preis der Woche erst mit Abschluss der Woche 1 gültig.

Ebenso sollten unterschiedlichen Intervalle jeweils ein Vielfaches voneinander betragen. So ist das Wochenintervall das 5-fache eines Tagesintervalls. Werden aber z.B. ein Wochenintervall und ein Monatsintervall zusammen verwendet, so entspricht u.a. der Schlusskurs des Monats nur dann dem Schlusskurs der letzten Woche des Monats, wenn dieser auch der letzte Tag des Monats ist.

TradeKnowledge führt deswegen eine Überprüfung der im Signalterm enthaltenen LHC-Preise und deren Intervalle durch und lässt Signalterme die LHC-Preise mit einem größeren Intervall als mit Szenario@OrderIntervall angegeben sind, nicht zu und gibt einen entsprechenden Hinweis aus. In unserem Beispiel wäre der Wert „Day“ für Szenario@OrderIntervall nicht möglich. Für Szenario@OrderIntervall sollte deshalb „Week“ gesetzt werden. Weiterhin wird bei Verwendung von unterschiedlichen Intervallen, die kein Vielfaches voneinander sind, eine entsprechende Warnung ausgegeben.

7.1.2.1.3 Intervall eines Charts

Wird das Handelssystem mit einem Chart verbunden (so für eine Visualisierung), so sollte der Chart mit dem kleinsten verwendeten OrderIntervall aller Szenarien des Handelssystems angezeigt werden. So können im Chart auch alle Informationen aus den Szenarien, mit kleineren Intervallen, angezeigt werden. Verwendet z.B. das Handelssystem für das Einstiegsszenario das Wochenintervall und für das Ausstiegsszenario das Tagesintervall, so sollte der Chart im Tagesintervall angezeigt werden.

TradeKnowledge liefert die Information, über das zu verwendende Intervall für den Chart.

7.1.2.1.4 Ausführung von gültigen Handelssignalen

Liegt ein gültiges Handelssignal vor, so muss eine entsprechende Ausführung des Handels erfolgen.

Dieses erfolgt durch die Generierung einer entsprechenden Orderanweisung (siehe 7.1.3). Z.Zt. unterstützt TradeKnowledge nur die einfachsten Ordertypen (wie Market, Stopp- und Limit).

Über das Attribut @OrderTimeframe werden für ein Szenario die ausführbaren Handelszeiten festgelegt. So ist es für einen Berufstätigen kaum möglich, während der öffentlichen Handelszeiten (so z.B. für XETRA

montags bis freitags von 9:00 bis 17:30 Uhr MEZ) Orders an der Börse zu platzieren. Mögliche Werte für OrderTimeframe sind z.Zt.

- **InsideTradingHours**
  Orderaufgaben innerhalb der Handelszeiten
- **OutsideTradingHours**
  Orderaufgaben außerhalb dieser Zeiten (außerbörslicher Handel oder ein anderer Börsenplatz)
- **AllTradingHours**
  Orderaufgaben inner- und außerhalb der Handelszeiten möglich

Über das Attribut @SignalTermUsage erfolgt für ein Szenario die Angabe, wie auf das Handelssignal reagiert werden soll. Mögliche Werte sind z.Zt.

- **AtOpenThisBar**
  Mit einem Handelssignal soll ein Handel zum Eröffnungskurs des aktuellen Intervalls erfolgen.
- **AtCloseThisBar**
  Mit einem Handelssignal soll ein Handel zum Schlusskurs des aktuellen Intervalls erfolgen.
- **AtLimitPrice**
  Mit einem Handelssignal soll ein Handel zum angegebenen Limit-Preis (Attributwert @SignalPrice) erfolgen.
- **AtStopPrice**
  Mit einem Handelssignal soll ein Handel zum angegebenen Stopp-Preis (Attributwert @SignalPrice) erfolgen.
- **AtSimPrice**
  Mit einem Handelssignal soll ein Handel zum angegebenen Preis (Attributwert @SignalPrice) erfolgen.

Hinweis: TradeKnowledge setzt @SignalPrice für AtOpenThisBar auf den Open- und für AtCloseThisBar auf den Close-Preis.

7.1.2.1.5 Vordefinierte Szenarien

TradeKnowledge bietet mit **DefaultExitLimitSzenario** ein vorgefertigtes Szenario für eine Risikobegrenzung an. Dieses generiert eine entsprechende Stopp-Loss-Order zeitnah nach einem Einstieg.

Mit diesem Szenario gilt:

- Als ExitSzenarioType ist LossLimitExit gesetzt. Damit wird das Szenario als Ausstiegsszenario für eine Risikobegrenzung gekennzeichnet.
- Für SignalTermUsage wird AtStopPrice verwendet.
- Für SignalPrice wird der Wert des Attributes „InitialStopPriceAtEntry“ (siehe 7.1.1) als Ausstiegspreis angegeben.
- Für SignalTerm wird der vorgefertigte Term **AtEntry** verwendet. Damit wird die Stopp-Order zeitnah nach einem Einstieg aufgegeben.

7.1.2.1.6 Kombination von Szenarien

Besteht das Handelssystem aus mehreren Ein- und/oder Ausstiegs-Szenarien, so werden diese innerhalb der Ausführung des Handelssystems sequenziell abgearbeitet. Szenarien erhalten Informationen über bestehende Positionen über die Funktionalität **DetermineMarketState** (siehe 7.1.1). D.h. besteht z.B. das Handelssystem aus mehreren Einstiegsszenarien, so wird mit dem ersten gültigen Handelssignal eine Position eröffnet. Weitere Einstiegsszenarien können so die Auswertung des SignalTerms bei einer bereits bestehenden Aktienposition vermeiden.

7.1.3 Modellklasse: Order

Mit einer Order erfolgt die Ausführung des Handelssignals, so z.B. die Eröffnung einer Aktienposition über eine Kauforder.

Eine **Order** ist ein direkter Bestandteil eines Szenarios und wird von dieser verwaltet.

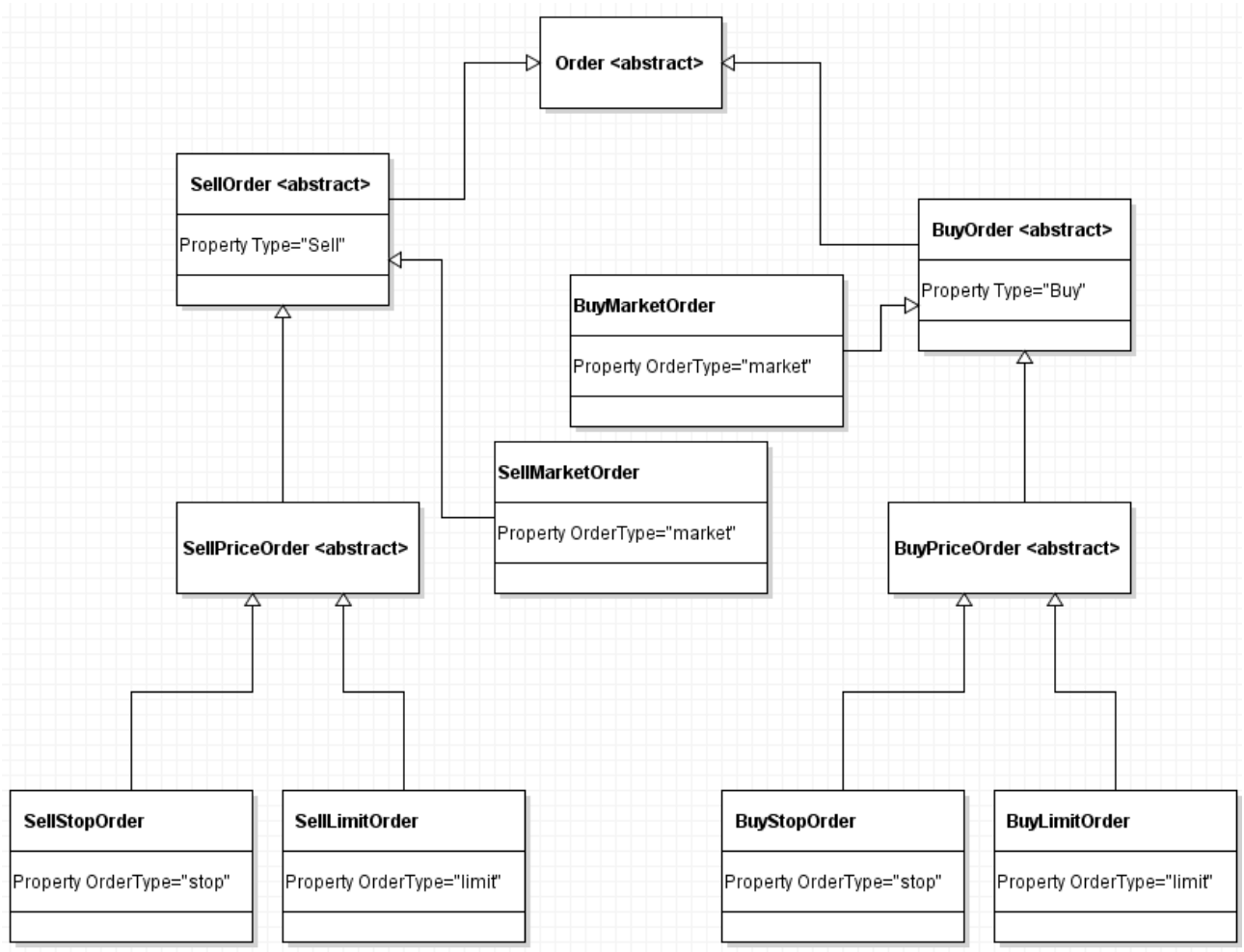

Abbildung 10: Modellstruktur Order

Für jeden Ordertyp gibt es in TradeKnowledge eine entsprechende Klasse zu deren Abbildung. TradeKnowledge verwendet die geeignete Orderklasse auf Basis, der in der im TradingSystem und im Szenario, festgelegten Eigenschaften.

So wird z.B. bei TradingSystem@TradeDirection=LONG und innerhalb eines Einstiegsszenarios eine Orderklassen für einen Kauf (Orderklasse mit dem Property #Type='Buy') verwendet.

Entsprechend des, im Szenario festgelegten, Wertes für @SignalTermUsage erfolgt die Verwendung einer entsprechenden Market, Stopp- oder Limit-Order.

TradeKnowledge betrachtet die folgenden Regeln:

- Bei TradingSystem@TradeDirection=LONG:
  - In einem Einstiegsszenario werden nur Kauf-Orders zur Eröffnung einer LONG-Position und bei einem Ausstiegsszenario werden entsprechende Verkaufsorders zur Schließung einer bestehenden LONG-Position ausgeführt.

    - In einem Einstiegsszenario wird nur eine Order ausgeführt, wenn für diese Aktie noch keine LONG-Position vorhanden ist (z.Zt. wird keine Pyramidisierung unterstützt). Die Information, ob bereits Positionen für eine Aktie vorhanden sind, wird über die Komponente **MarketState** innerhalb der Komponente **PositionsManagement** (siehe 7.1.1) bestimmt.
    - Analog wird in einem Ausstiegsszenario nur eine Order ausgeführt, wenn für diese Aktie entsprechend eine LONG-Position vorhanden ist.
    - Befindet sich der Einstiegskurs oberhalb des initial Stopp-Preises (siehe RisikoAndMoneyManagememt/InitialStopPriceAtEntry).
- Analoges gilt entsprechend bei Tradingsystem@TradeDirection=SHORT für SHORT-Positionen (z.Zt. durch TradeKnowledge jedoch nicht unterstützt).
- In einem Einstiegsszenario wird die Positionsgröße (also die Anzahl zu kaufender Aktien) innerhalb der Komponente **DetermineStockAmount** (siehe 7.1.1) bestimmt.
- Bei Tradingsystem@OrderRouting=indirectOrderRouting werden innerhalb des produktiven Programmcodes keine Orderanweisungen generiert. Es werden aber zusätzliche Meldungen generiert, die die Informationen zu einer manuellen Ordereingabe beim Broker enthalten. Der Programmcode zur Ausführung von Backtests enthält aber entsprechende Orderanweisungen, um entsprechende Simulationen ausführen lassen zu können.

In einer weiteren Ausbaustufe von TradeKnowledge können noch weitere Orderarten und ein Abgleich mit einem entsprechenden Brokerprofil, bei dem die unterstützten Orders beschrieben werden, erfolgen.

7.2 TRADER-PROFIL

Jedes Handelssystem muss für seinen vorhergesehenen Anwendungsfall geeignet sein. So ist für einen Berufstätigen (der z.B. 40 Stunden in der Woche im Büro sitzt) ein Handelssystem im 1h-Chart und mit der Notwendigkeit der manuellen Ordereingabe völlig ungeeignet.

Durch die Definition von entsprechenden Profilen für den Anwender eines Handelssysteme kann TradeKnowledge für diese Anwender geeignete Attributwerte verwenden und entsprechende Vorschläge zu deren Verwendung erzeugen. Die Verwendung ungeeigneter Werte führen zu einem nicht gültigen Modell.

TradeKnowledge enthält z.Zt. die vordefinierten Trader-Profile „Employed“ und „Autonom“. Unser Berufstätiger würde also das Trade-Profil „Employed“ verwenden. In TradeKnowledge können Profile entsprechende erweitert oder neue Profile hinzugefügt werden.

Im Anhang wird die Verwendung weiterer Profilearten (so z.B. für den verwendeten Broker oder Handelsumgebung; siehe Anhang, Kapitel 7.1.) vorgestellt.

7.2.1 Trader-Profil „Employed“

Ein voll Berufstätiger mit einer 40-Stunden Arbeitswoche, der in seiner freien Zeit Aktien als LONG-Positionen handelt. Dieser Trader hat nur innerhalb seiner Freizeit, also nach Feierabend oder am Wochenende, Möglichkeiten die Signale des Handelssystems zu analysieren und Orders an seinen Broker weiterzuleiten. Ebenso hat dieser nur einen privaten Rechner zur Ausführung des Handelssystems und gewöhnlich keine API-Anbindung zu seinem Broker.

<u>Geeignete Werte:</u>

- TradingSystem@TradeDirection = LONG, für LONG-Positionen von z.B. Aktien
- TradingSystem@SignalRecognition = temporarySignalRecognition, für eine manuelle Ausführung des Handelssystems, d.h. das Tool z.B. AmiBroker wird gestartet und der Programmcode des Handelssystems ausgeführt
- TradingSystem@OrderRouting = indirectOrderRouting, für eine manuelle Ordereingabe entsprechend, dem von Handelssystem gelieferten Signale über die Ordermaske des Brokers
- Szenario@OrderInterval = Week, für einen Handel auf Wochenbasis.

- Szenario@OrderTimeFrame = OutsideTradingHours, für eine mögliche Orderaufgaben außerhalb der Börsenöffnungszeiten (außerbörslicher Handel oder ein anderer Börsenplatz)
- Tradingsystem@onlyBacktest = False, da auch real gehandelt werden soll

Ungeeignete Werte:

- Szenario@OrderInterval = kleiner Day (z.B. Hour), Handel im Intradaybereich nicht möglich

7.2.2 Trader-Profil „Autonom“

Das Handelssystems läuft in einer sicheren Laufzeit-Umgebung rund um die Uhr und besitzt eine API zum Broker. Somit können jederzeit und zeitnah Handelssignale ausgewertet und Orders an den Broker übermittelt werden.

Empfohlene Werte:

- TradingSystem@SignalRecognition = permamentSignalRecognition, für eine stätige Auswertung von Handelssignalen.
- TradingSystem@OrderRouting = directOrderRouting, für ein direktes Orderrouting an den Broker (z.B. über eine entsprechende API).
- Szenario@OrderTimeFrame = InsideTradingHours, für eine mögliche Orderaufgabe innerhalb der Börsenöffnungszeiten.
- Tradingsystem@onlyBacktest = False, da auch real gehandelt werden soll.

7.3 MELDUNGEN VON TRADEKNOWLEDGE

Durch Handelsalgorithmen können bestimmte und u.U. ungünstige Ereignisse auftreten. Auf Basis von vorhandenem Wissen, Erfahrungen und Best-Practices kann TradeKnowledge entsprechende Meldungen und Hinweise erzeugen.

7.3.1 Meldungen innerhalb der Modellierungsphase

In TradeKnowledge können jedem Zustand des Modells, also die jeweils aktuelle Zusammensetzung von Instanzen und Attributwerte, Meldungen zugewiesen werden. Somit erhält der Handelssystementwickler während der Modellierung entsprechende Hinweise, Vorschläge, Empfehlungen und Warnungen und kann so das Modell entsprechend korrigieren oder andere Entscheidungen treffen. Dadurch wird der Entwicklungsprozess interaktiv unterstützt, der Entwickler geleitet und ggf. auf neue Ideen gebracht.

So würde z.B. bei einem gesetzten Attributwert ‚permanentSignalRecognition‘ für @SignalRecognition innerhalb der TradingSystem-Instanz der folgende Vorschlag angezeigt: *Für das „Attribut @OrderRouting sollte ‚directOrderRouting‘ gesetzt werden.“* D.h. das bei einer dauerhaften Auswertung von Börsendaten und die damit verbundene zeitnahe Verfügbarkeit von Handelssignalen eine direkte und automatisierte Orderübergabe an den Broker sinnvoll ist.

Weitere Beispiele für Meldungen werden u.a. im Kapitel 7.4 und im Anhang zu dieser Arbeit (Kapitel 6.3.1) gelistet.

In weiteren Versionen von TradeKnowledge könnten ebenso Vorschläge für geeignete und ggf. vordefinierte Ein- und Ausstiegsszenarien und Indikatoren als Meldungen ausgegeben werden.

7.3.2 Meldungen zur Laufzeit

Gewisse Zustände sind vom aktuellen Markt abhängig (so z.B. vom jeweils aktuellen Preis) und können so nur innerhalb der Laufzeit, also während eines Backtests oder während des praktischen Handels erkannt werden. Das Erkennen solcher Zustände ist nur über, innerhalb des Programmcodes enthaltenen Prüfmethoden möglich. Diese Prüfmethoden werden dem Code innerhalb der Transformationsphase hinzugefügt und erzeugen zur Laufzeit entsprechende Meldungen. Auf Basis der Meldungen, die während der Backtests oder zur

Laufzeit entstanden sind, kann der Handelssystementwickler das Modell entsprechend korrigieren und verbessern.

So wird z.B. während des Backtest erkannt, dass zum selben Bar ein Handelssignal für einen Einstieg und ebenso ein Handelssignal für einen Ausstieg gültig sind. Mit einer entsprechenden Warnung kann der Entwickler die Signalterme entsprechend korrigieren.

Weitere Beispiele für Meldungen werden im Anhang zu dieser Arbeit (Kapitel 6.3.2) gelistet.

7.4 HANDELBARKEIT UND PRAKTISCHE ANWENDBARKEIT

Ein wesentlicher Aspekt bei der Entwicklung von Handelssystemen ist die praktische Anwendbarkeit mit realem Geld. Also kann das Handelssystem unter den persönlichen, finanziellen, zeitlichen und sonstigen Gegebenheiten und Umständen praktisch umgesetzt und gehandelt werden?

Entsprechend der Gültigkeit von Preisen innerhalb eines Intervalls (siehe 7.1.2.1.1), der Auswertung von Signalen und der Art der Orderaufgabe (siehe 7.1.3) sowie weiteren Aspekten beeinflussen die Handelbarkeit eines Handelssystems und damit die praktische Anwendbarkeit.

Im Folgenden werden verschiedene Zeitpunkte der Auswertung des Signalterms betrachtet. Hieraus schließt TradeKnowledge die Gültigkeit von, innerhalb des Signalterms enthaltenen, Preise und möglichen Arten der Orderanweisungen.

Die folgende Tabelle zeigt die Zeitpunkte der Auswertung des Signalterms, dessen mögliche Kombinationen von Attributwerten und Meldungen.

Fall	**Zeitpunkt Auswertung Signalterm**	**Signalterm darf enthalten**	**Szenarios@Signal-TermUsage**	**Szenarios@ Order-TimeFrame**	**TradingSys-tem@ SignalRecogni-tion**	**TradingSys-tem@ OrderRouting**
A1	Beginn des aktuellen Intervalls (erster Tick)	Open aktuelles Intervall und alle historische OLHC	AtOpenThisBar	Inside / All	Permament	Direct
A2			AtLimit, AtStop	Inside / All	Permament	Direct / Indirect
B1	Innerhalb des aktuellen Intervalls		AtMarketImmediately (z.Zt. nicht unterstützt)	Inside / All	Permament	Direct
B2			AtLimit, AtStop	Inside / All	Permament / Temporary	Direct / Indirect
C1	Ende des aktuellen Intervalls (letzter Tick).	OLHC aktuell abgeschlossenes Intervall und alle historische OLHC	AtCloseThisBar	Outside / All	Permament	Direct
C2			AtLimit, AtStop	Outside / All	Permament	Direct
D1	Zwischen Handelsende und vor nächster Handelseröffnung		AtCloseThisBar	Outside / All	Permament / Temporary	Direct / Indirect
D2			AtLimit, AtStop	Outside / All	Permament / Temporary	Direct / Indirect
E1	Mit Beginn der nächsten Handelseröffnung	Bereits neues Intervall gültig				

Tabelle 1: Anwendbarkeit von Signaltermen

Meldungen (siehe 7.3.1):

A1) „Eine Ausführung ist nur zum ungefähren Eröffnungskurs möglich."

C1 und D1) „Eine Ausführung ist nur zum ungefähren Schlusskurs außerhalb der Handelszeiten möglich. Alternativ sollte eine Limit-Order verwendet werden."

Für alle AtStop: „Eine Ausführung der Order wird nicht garantiert. Ebenso kann die Order mit einem ungünstigeren Preis ausgeführt werden"

Für alle AtLimit: „Eine Ausführung der Order wird nicht garantiert."

Fußnoten:

A1) Erster Tickpreis (Open) des Intervalls kann nur mit nachfolgenden Ticks gehandelt werden. Der ausgeführte Preis kann sich vom Open-Preis unterscheiden.

B1) TradeKnowledge schließt Tickpreise innerhalb des SignalTerms aus (siehe 7.1.2.1.1). So kann eine sofortige Market-Order innerhalb eines Intervalls nur durch andere Events (so z.B. zeitliche Auslöser oder Nachrichten) einen gültiges Handelssignal verursachen. Dieses wird jedoch z.Zt. nicht unterstützt.

C1) Letzter Tickpreis (Close) des Intervalls kann nur mit nachfolgenden Ticks außerbörslich bzw. an anderen Börsen gehandelt werden. Der ausgeführte Preis kann sich vom Close-Preis unterscheiden.

E1) Handelssysteme, die eine Orderausführung zum jeweils nächsten Bar verwenden, werden durch TradeKnowledge nicht unterstützt. So z.B. eröffne eine Position zum Open des nächsten Bars. Modelle sollten in diesem Fall die Auswertung des SignalTerms auf historische Intervalle anwenden. Im Beispiel das vorherige Intervall und Fall A1.

Für alle historischen OLHC-Preise gilt, dass diese mit AtStop und AtLimit gehandelt werden können.

Betrachten wir die Verwendung des Trader-Profils „Employed". Hier wären durch die Werte TradingSystem@SignalRecognition=temporarySignalRecognition, TradingSystem@OrderRouting=indirectOrderRouting und Szenario@OrderTimeFrame=OutsideTradingHours nur der Fall D mit Szenarios@SignalTermUsage = AtCloseThisBar, AtLimit oder AtStop möglich.

Eine Kombination von TradingSystem@SignalRecognition=directSignalRecognition und TradingSystem@OrderRouting=directOrderRouting ist z.B. notwendig, wenn sofort auf Marktänderungen reagiert werden und eine direkte und umgehende Orderaufgabe erfolgen soll. So z.B. im Fall A1 für eine zeitnahe Ausführung AtMarket zum ungefähren Einstiegspreis. Bei anderen Kombinationen würden Handelssignale zu spät erkannt (mit TradingSystem@SignalRecognition=indirectSignalRecognition) oder Orders erst verzögert an den Broker weitergeleitet (mit TradingSystem@OrderRouting=indirectOrderRouting). So kann sich der Preis am Markt bereits ungünstig verändert haben.

7.5 VERMEIDUNG TYPISCHER FEHLER IN DER HANDELSSYSTEMENTWICKLUNG

Mit dem Wissen von typischen Fehlern, die in der Handelssystementwicklung auftreten können, können mit TradeKnowledge entsprechende Regeln zu deren Reduzierung oder Vermeidung definiert werden.

Die folgende Aufzählung typischer Fehler und deren Behandlung durch TradeKnowledge kann ständig erweitert werden:

7.5.1 Einstiegspreis zu nah am Stoppkurs

Liegt der Einstiegspreis zu dicht am initialen Stoppkurs, so besteht die Gefahr eines frühzeitigen Ausstoppens der Position. Außerdem kann durch die zu geringe Preisspanne ein überhöhter Kapitaleinsatz für die Position entstehen.

Durch eine festgelegte Mindestspanne (in Prozent) über das Attribut RisikoAndMoneyManagememt@MinimumDistanceFromInitialStopPriceAtEntry werden Handelssignale mit einer zu geringen Spanne ignoriert.

7.5.2 Annahme ausgeführter Stopp- und Limit-Order

Bei der Verwendung von Stopp- und Limit-Ordern in Backtests werden im Backtest oft die angegebenen Preise als ausgeführt angenommen. Hierbei erfolgt die Annahme, dass alle Preise zwischen Low und High des Intervalls als handelbar gelten. Für eine naheliegende Simulation von Stopp- und Limit-Orders müssten alle historischen Tickpreise eines Intervalls verwendet werden; was aber in der Praxis bzw. Datenlieferung und Datenmenge schwer möglich ist.

Ebenso muss in der Praxis meist der Orderpreis einer Limit-Order über- bzw. unterstiegen werden, damit die Order überhaupt ausgeführt wird.

Bei einer Stopp-Order wird mit Erreichen des Order-Preises eine Market-Order ausgelöst. Hierdurch kann ein viel ungünstigerer Preis gehandelt werden.

TradeKnowledge erlaubt mit Szenario@SignalTermUsage=AtStop und AtLimit die Verwendung einer Stopp- und Limit-Order. Während der Modellierung erzeugt TradeKnowledge entsprechende Meldungen, bei deren Verwendung (siehe 7.4). Für den ausführbaren Programmcode generiert TradeKnowledge die für die Programmiersprache vorgesehenen Stopp- und Limit-Orderanweisungen und nutzt somit deren Standardverhalten.

In weiteren Versionen von TradeKnowledge wären die folgenden Erweiterungen für Backtests denkbar:

- Verwendung der niedrigsten verfügbaren Zeitebene (ggf. Tickpreise).
- Annähernde Simulation von Tickpreisen durch die Definition einer Tickgröße, welches den für den Markt üblichen Abstand zwischen zwei Tickpreisen angibt (z.B. 0.01 für Aktien). So z.B. mit der Variable TickSize in AFL für AmiBroker

7.5.3 Annahme Ausführung zum Schluss-Kurs

In Backtests wird oft der Schlusskurs eines Intervalls als Preis für eine Ausführung verwendet (so z.B.: Kaufe Aktien zum Schlusskurs des Tages). Da der Schlusskurs eines Intervalls dessen letzten Tickpreis entspricht, ist im realen Handel eine Ausführung genau zu diesem Preis praktisch unmöglich.

Es muss also mit einer Ausführung zu einem annähernden Preis gerechnet werden und diese Toleranz im Backtest mit einberechnet werden. Der Marktpreis zum Zeitpunkt der Ordereingabe kann sich sogar stark vom Schlusskurs unterscheiden, so dass die Order ggf. nur zu einem viel ungünstigeren Kurs ausgeführt wird.

Kapitel 7.4 zeigt wie TradeKnowledge in diesem Fall entsprechende Meldungen und Hinweise generiert und entsprechend dem Anwendungsfall des Handelssystems nur sinnvolle Orderanweisungen zulässt.

In weiteren Versionen von TradeKnowledge wären die folgenden Erweiterungen für Backtests denkbar:

- Ermittlung eines für den Markt üblichen Toleranzwertes und Einberechnung in den Backtest. So kann der Schlusskurs entsprechend der Toleranz „schlechter“ gerechnet werden. Dieser Toleranzwert könnte bei einer manuellen Ordereingabe dem Anwender als Hinweis ausgegeben werden.
- Der Backtest liefert zusätzlich, z.B. auf Basis des Toleranzwertes, einen Konfidenzintervall, welcher die Unterschiede zwischen Backtest und realitätsnaher Ausführung wiedergibt.

7.5.4 Verwendung nicht handelbarer Preise in einer Order

Einige Software für Backtests erlauben die Angaben von direkten Preisen in einer Order, zu der diese dann ausgeführt wird. So z.B. mit der Variable BuyPrice in AFL für AmiBroker.

Diese Orderanweisungen sind jedoch real nicht handelbar und machen nur zu Analysezwecken Sinn. So um z.B. eine Strategie mit dem ungünstigsten Ausstieg aus einer LONG-Position innerhalb eines Intervalls mit der Angabe des Low des aktuellen Intervalls zu evaluieren. Ebenso kann mit solch einer Order ein ungewollter Zukunftsblick im Backtest entstehen, so mit der Angabe eines LHC-Preises eines höheren Intervalls (siehe 7.1.2.1.2).

TradeKnowledge erlaubt mit Szenario@SignalTermUsage=AtSimPrice die Verwendung von Orders mit direkter Preisangabe. Dieses ist jedoch nur möglich mit TradingSystem@onlyBacktest=True; also nur zu Analysezwecken. Aus solch einem Modell kann kein Programcode für einen realen Einsatz generiert werden.

Soll ein Handelssystem real eingesetzt werden, so muss TradingSystem@onlyBacktest=False gesetzt werden. Im Backtest sind dann Orders mit Preisangabe nicht möglich und können nicht zu ungewollten Zukunftsblicken oder nicht handelbaren Orders führen.

7.5.5 Zukunftsblick bei Backtests

Während eines Backtests werden durch die Intervalle, für die historische Daten zur Verfügung stehen, iteriert. Es wird also die Vergangenheit von frühester Zeit bis zur Gegenwart durchgespielt und ein Tradeverlauf simuliert.

Dabei dürfen keine historischen Werte verwendet werden die, ausgehend vom gerade aktuellen simulierten Intervall, in der Zukunft liegen. Wird z.B. der Handel vom 14.06.2021 simuliert, so dürfen keine Daten nach dem 14.06.2021 verwendet werden. Durch Unwissenheit und Fehler in der Programmierung kann jedoch ein solcher Zugriff entstehen. Eine Verwendung von Daten aus der Zukunft verursachen einen Zukunftsblick im Backtest. Die Ergebnisse im Backtestreport sind unrealistisch und die Orders real nicht handelbar.

Zugriffe auf zukünftige Daten sollten von der entsprechenden Backtest-Software erkannt, verhindert und als Fehler gemeldet werden. Doch trotz aller Sorgfalt in der Entwicklung können Zukunftsblicke entstehen.

Sie können gewollt sein, z.B. für Analysezwecke (etwa durch Indikatoren, die auf eine vorrausschauende Berechnung basieren, z.B. der ZigZag-Indikator).

Beispiele:

- In einem Handelssystem wird die Variable ‚histBars' als Index für eine Datenreihe verwendet. So liefert Close[histBars] mit histBars=1 den Schlusskurs des letzten Intervalls. Durch einen Fehler im Algorithmus werden aber auch negative Werte für histBars berechnet. Somit werden auf Werte aus der Zukunft zugegriffen und diese ggf. im Backtest verwendet.
- Ein Handelssystementwickler verwendet den ZigZag-Indikator, um entsprechende Kauf-Signale für Aktien zu generieren. Ihm ist aber nicht bewusst, dass dieser Indikator ein vorrausschauender Indikator ist. Im Backtest wird eine „traumhafte" Equity angezeigt.

Es gilt also, ungewollte Zukunftsblicke in Algorithmen frühestmöglich zu erkennen und diese zu vermeiden.

Vollständig lässt sich die Aufdeckung eines Zukunftsblicks durch die automatisierte Interpretation von TradeKnowledge z.Zt. nicht realisieren. Dazu müsste jeder verwendete Algorithmus, und auch der von Drittanbietern, auf die Verwendung von Zukunftsblicken automatisiert überprüft und analysiert werden können.

TradeKnowledge bietet zwei alternative Ansätze an, die im Anhang zu dieser Arbeit im Kapitel 7.7. detailliert beschrieben werden:

7.5.5.1 Überprüfung und Zertifizierung von Termen

Hierbei erfolgt eine explizite Überprüfung und Einstufung von Termen bzgl. des Zukunftsblickes. Auf Basis dieser erfolgt eine entsprechende Modellvalidierung.

7.5.5.2 2-Phasen Modellprozessierung

Hierbei erfolgt eine Ausführung von zwei unterschiedlichen Backtestläufen auf das gleiche Modell. Liefert der Backtestlauf mit den originalen historischen Daten das gleiche Ergebnis, wie der Backtestlauf mit modifizierten (für jeden Intervall jeweils in der Zukunft liegenden) historischen Daten, so kann von einem zukunftsfreien Blick ausgegangen werden.

8 ANHANG

Der Anhang zu dieser Arbeit, als separates Dokument, enthält zusätzliche Informationen zur Wissensbeschreibung, weiteren Bestandteilen zu TradeKnowledge und weiterführende Ideen. Es wird eine prototypische Realisierung vorgestellt, mit der der vorgestellte Ansatz von TradeKnowledge praktisch evaluiert wurde. Es folgt eine beispielhafte Modellierung eines einfachen Handelssystems.

9 SCHLUSSWORT

Mit dieser Arbeit wurde ein Konzept zur Entwicklung von Handelssystemen vorgesellt. Dieses Konzept beruht auf der Produktkonfiguration und das Produkt ist der lauffähige Programmcode eines Handelssystems.

Mit TradeKnowledge ergeben sich neue interessante und höchstwahrscheinlich noch nicht erkannten Möglichkeiten. Es ist offen für Korrekturen und Erweiterungen. Es sollte bei Interesse veröffentlicht und innerhalb einer breiten Community von Tradern, Handelssystementwickler, Firmen von Handelssoftware und Brokern weitergetrieben und gepflegt werden. So kann das breite Wissen und die Erfahrungen gesammelt und für alle nutzbar gemacht werden.

Ich, als Autor, bin von meiner Idee und den beschriebenen Einsatzmöglichkeiten überzeugt. Ich hoffe auf Akzeptanz und Unterstützung bei der weiteren Umsetzung. Vielen Dank.