Proceedings of the 7th WSEAS International Conference on Mathematics & Computers in Business & Economics, Cavtat, Croatia, June 13-15, 2006 (pp13-18)

Derivative Contracts as Active Documents

Component-Orientation meets Financial Modeling

Markus Reitz*
University of Kaiserslautern
Software Technology Group
P.O. Box 3049, 67653 Kaiserslautern
Germany

Ulrich Nögel*
Fraunhofer ITWM
Department of Financial Mathematics
Fraunhofer-Platz 1, 67663 Kaiserslautern
Germany

Abstract: Derivative contracts represent a very important and constantly growing financial segment of local and world-wide markets. To assure or even improve one's position in an environment characterised by permanently shortening product life and time to market cycles, global as well as local market players have to optimise the overall product design process. The inherent flexibility of derivatives creates both: nearly unbounded opportunities for new and innovative contracts and the pressure to use efficient methods for contract composition, processing, management and valuation. Techniques based on the concept of an ACTIVE DOCUMENT, a component-oriented approach, are able to offer novel solutions for these problems. This paper discusses ACTIVE DOCUMENTS in the context of derivatives, sketches possible benefits when compared to state of the practice techniques and outlines new scenarios for ingenious usage of the added value of ACTIVE DOCUMENT systems.

Key–Words: Active Documents, Derivative Contracts, Component-Orientation, Financial Engineering, Java, XML

1 Introduction

To cope with the problems resulting from permanently shortening product development cycles, financial engineers need new design methodologies and software-based supportive technologies. COMDECO[1] aims at providing means to generically compose, check and store derivative contracts, using an ACTIVE DOCUMENT based representation technique whose level of abstraction is above that of plain mathematical formulae. Based on this abstraction, structural as well as semantical consistency checks are available, freeing the financial engineer from mind-numbing tasks, letting him concentrate on the really creative aspects of the design process. Besides those checks, the additional information provided by this abstraction may be used to automatically construct effective valuation algorithms for derivative contracts, too. Depending on the specific contract, the supporting framework is able to select the best matching valuation algorithm for fair price calculations. To be of practical relevance, the approach has to offer both:

1. **Backwards Compatibility**, i.e. existing derivatives should be expressible in terms of the approach, and

2. **Openness**, i.e. properties and features of next generation derivatives should be modelable effectively.

Component-oriented technologies, manifested in the ACTIVE DOCUMENT framework, are applied to the domain of financial modeling, transferring their immanent flexibility to derivative contracts. Any derivative contract may be expressed in terms of such a document, whereas in this context, the term *active* refers to

- the ability to automatically *check and enforce* consistency constraints, i.e. avoid invalid or meaningless contracts.

- the ability to automatically *perform context-specific adaptations*, e.g. find the best-matching valuation algorithm.

This paper introduces the overall concepts of COMDECO, gives an overview of the project and sketches questions that will be investigated in forthcoming working packages.

*Supported by the cluster of excellence *Dependable Adaptive Systems and Mathematical Modeling* (DASMOD) of Rhineland-Palatinate, Germany.

[1]Composable Derivative Contracts, a subproject of DASMOD (`http://www.dasmod.de`).

2 Related Work

ACTIVE DOCUMENTS are an evolutionary continuation of *compound document* technologies. Introduced with Microsoft OLE and further refined in COM [1], a compound document is able to combine different media data types and allows for *in-place editing*. Apple's OpenDoc [2] aimed at providing a multi-platform framework, but the attempt was not crowned with success. Starting with Mac OS X, OpenDoc has been declared a deprecated technology and is no longer supported. *Minerva* [3] introduced the notion of an ACTIVE DOCUMENT in the context of e-Learning during the Easycomp project [4]. COMDECO as well as the underlying framework for general-purpose ACTIVE DOCUMENTS are based on the insights and results of this EU-funded research.

3 ACTIVE DOCUMENTS

When modeling derivative contracts, the general-purpose framework is specialised towards a specific variant, *Hierarchical* ACTIVE DOCUMENTS, which is used for contract representation by the runtime system. Being hierarchical, derivative contracts and their corresponding ACTIVE DOCUMENT representation may be expressed in a XML-based format.

Example 1 *Figure 1 represents a derivative contract C having a payoff of*

$$P_C(t) = \begin{cases} \min(\max(FOO(t) - 12, 10), 20) & t = 7 \\ \textbf{unknown} & \text{otherwise} \end{cases}$$

The transformation from P_C to $\hat{P}_C$, so

$$\forall t \leq \text{MATURITY(C)} : \hat{P}_C(t) \neq \textbf{unknown}$$

holds, is performed by an appropriate valuation process whose internals are described in Section 4.

For *Hierarchical* ACTIVE DOCUMENTS, additional constraints for the principal entities *environment* and *component* have to be taken into account:

- An environment embeds at most one component and an arbitrary number of environments.
- Environments enforce the *local messages only* constraint, i.e. only intra-environmental messages may pass environmental boundaries.

Using XML as an intermediate representation, document interchanging and transformation could be performed using standard technologies, e.g. *XML Schema* or *XSL*. Besides graphical derivative development environments, simple text editors may be used to handle contracts, because of a textual representation.

Mathematical Assembler Implementing a contract with the help of mathematical formulae is similar to programming in low-level assembler, missing all the bells and whistles of modern programming languages, e.g. types. Moving from a hard-coded implementation based on pure mathematical descriptions to ACTIVE DOCUMENTS is comparable to the transition from assembly language to a high-level programming language. The supporting framework provides similar or analogous features a conventional IDE offers to its users[2].

3.1 Advantages

For decades, financial engineers have designed and analysed derivative contracts using plain mathematical formulae and have done well using straightforward description techniques - so why switch over to ACTIVE DOCUMENTS? The answer is quite simple: With increasing complexity and decreasing turnaround time, current low-level description techniques do not scale up well, limiting the set of handleable contracts and usage scenarios.

Being a loosely coupled system, ACTIVE DOCUMENTS offer a high degree of flexibility and extensibility. For example, to support smoothing based on the geometric mean besides the already available smoothing using the arithmetic mean, adding a specific component is sufficient. All parts of the system, e.g. valuation and composition facilities, are made aware of the improved feature set automatically.

Because of component-orientation, every user is able to configure the system according to specific needs by adapting the general-purpose system adequately. Reinventing the wheel for every contract by specifically developing composition and valuation facilities for each category is unnecessary because of the provided genericity. System adaptation according to individual user requirements is therefore simple.

Structural and semantical checking facilities optimise the design process and help to evaluate a contract in its early design stage using explorative composition techniques.

3.2 From simple to complex contracts

Any derivative contract can be defined by its payoff and additional information concerning the conditions that allow to exercise it, e.g. *european* or *american style* derivatives. Because of the tremendous high degree of flexibility[3], a payoff may be as simple as in

[2] From the perspective of a general-purpose ACTIVE DOCUMENT framework, an IDE is a special kind of ACTIVE DOCUMENT, too.

[3] "With derivatives you can have almost any payoff pattern you want. If you can draw it on paper, or describe it in words, someone

```
<contract>
  <cap using="upperBound">
    <constant id="upperBound" value="20"/>
    <floor using="lowerBound">
      <constant id="lowerBound" value="10"/>
      <derivative>
        <sell>
          <condition>
            <at timestep="7"/>
          </condition>
          <observable id="foo" model="FOO"/>
        </sell>
        <acquire>
          <condition>
            <at timestep="0"/>
          </condition>
          <constant id="strike" value="12"/>
        </acquire>
      </derivative>
    </floor>
  </cap>
</contract>
```

Figure 1: A serialised contract representing a derivative based on the underlying FOO*, whose value is limited by an upper bound of 20 and a lower bound of 10 monetary units. The underlying is* acquired virtually *at timestep 0 and* sold virtually *at timestep 7.*

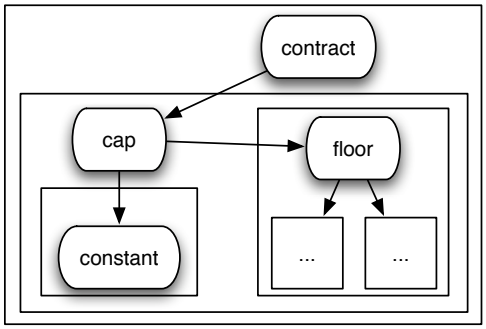

Figure 2: Instantiation of the contract on the left (boxes represent environments). Because of the local messages only *constraint, a specific message propagation scheme, e.g. only messages from the* contract *component reach the* cap *component, is guaranteed by the runtime system.*

the case of a plain vanilla european call option having a strike of K and maturity T

$$P(T) = \max(S(T) - K, 0)$$

or as complex as in the case of a globally capped and floored cliquet ($0 \leq t_1 \leq t_2 \cdots \leq t_n = T$)

$$P(T) = \left(\sum_{i=1}^{n} \max \left(\min \left(\frac{S(t_i) - S(t_{i-1})}{S(t_i)}, F \right), C \right) \right)^+$$

Cliquet options belong to the class of structured options, being part of a new generation of highly complex OTC[4] products, whose importance will increase in the future. Construction and pricing of this and other classes of options are complex and tedious tasks that are simplified by Active Document technologies.

3.3 Contract Composition

Active Documents may be instantiated from a valid XML description, but when designing new contracts, interactive and feedback-driven design techniques are superior to their non-interactive counterparts. An explorative design style is supported and enforced, creating the need for a (at least) three-valued classification scheme in contrast to non-interactive design processes.

Consistent States represent all circumstances, which are valid according to the composition specification.

Transitional States represent all situations, in which the composition specification is violated, but certain constructive, i.e. additive, operations performed by the Active Document would result in a consistent state.

Inconsistent States represent all circumstances, which are invalid according to the composition specification. In inconsistent state, only destructive operations, i.e. operations that remove parts of the Active Document, are allowed, as these lead to transitional or consistent states.

As the runtime system enforces the composition specification, it is impossible to transform a valid, i.e. consistent or transitional Active Document into an inconsistent one (see Figure 3). Directly manipulating serialised contracts, i.e. circumventing constraint checking, is the only possibility to create inconsistent Active Documents, leading to constraint violations during the instantiation phase.

3.3.1 Composition Constraints

By letting only reasonable contracts pass, composition constraints inhibit invalid and counterproductive composition attempts. Using higher level description techniques, many kinds of inconsistencies can be easily detected.

Example 2 *The payoff of a contract based on the given example with values of upper and lower bound accidently interchanged would be*

$$P_C(T) = \min(\max(FOO(T) - 12, 20), 10) = 10$$

can design a derivative that gives you that payoff." (Fischer Black, 1995)

[4]Over The Counter, i.e. a derivative is *specifically tailored* according to *individual customer requirements.*

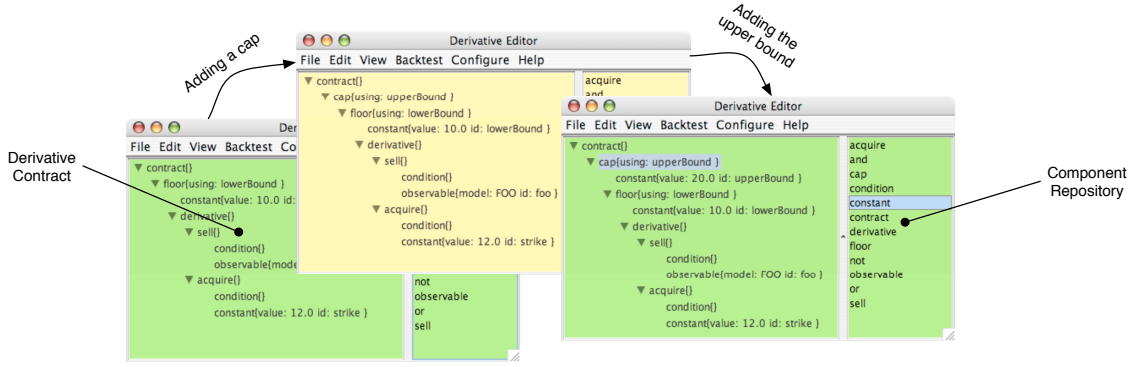

Figure 3: A contract in a consistent state is signaled by a green background (depending on the context, an empty condition defaults to timestep T or 0). Starting from a floored contract, e.g. made available by a contract repository, a cap component is added using an appropriate drag and drop gesture. The resulting contract is in transitional state, indicated by a yellow background. By defining the cap's upper bound, the contract is completed, leading to a consistent state again.

which is independent from the underlying's performance and in almost any case not the intended design.

Example 3 *An unintentionally double-floored contract could be detected and marked as erroneous or transformed to the optimised version*

$$\text{lowerBound} = \max(\text{lowerBound}_1, \text{lowerBound}_2)$$

Keep in mind that these examples are simple for illustration purposes only; real world contracts may exhibit a high degree of complexity, making general-purpose semantic checking facilities a valuable property of Active Documents. With ComDeCo also targeting the non-expert user as a contract designer (see Section 5), semantic checking becomes an essential aspect of the composition tool-chain.

3.4 Component Repository

The lynchpin of a derivative contract composition system is its component repository, which subsumes all building blocks available for contract design. By combining these pieces, the user is able to create consistent and semantical meaningful derivative contracts. As an essential aspect of an Active Document system is its openness with respect to future extensions, it is impossible to provide static composition specifications. Instead, a derivative contract component has to provide modular specification information, which describe the component's relation to other entities in a local as well as in a system-wide manner.

By combining these results, the runtime system is able to create a finalised document specification out of all available specification fragments. It should be obvious that the result of this process is dependent on the current component repository configuration, i.e. a customised specification is created.

4 Contract Valuation

Fair price calculations in conjunction with risk management decisions for derivative securities represent two of the most important topics in modern finance. Common mathematical approaches for solving the pricing problem can be assigned to one of three categories.

1. **Closed-form Solutions** provide analytical fair price formulae for a limited set of types of derivatives and models (see e.g. [5]). The lack of a general-purpose framework of closed-form solutions is compensated by the fact that only a minimal amount of computational resources is usually required.

2. **Monte Carlo Simulations** represent a nearly general-purpose and easy to understand approach for pricing a large amount of types of derivatives. Unfortunately, simulation is computationally intensive and does not provide a straightforward possibility to price american style derivatives.

3. **Tree-based Algorithms** allow for price calculations for european *and* american style derivatives, therefore offering a general-purpose framework for contract valuation. In contrast to Monte Carlo simulations, tree algorithms usually show up fast convergence, but in multi-dimensional settings, the memory footprint may impose limitations.

Categorisation scheme To provide a flexible and efficient valuation engine, a categorisation scheme, which is able to detect the applicability of closed-form solutions when appropriate, use tree-based algorithms in the common case and provide Monte Carlo simulations as an additional method, has to be developed.

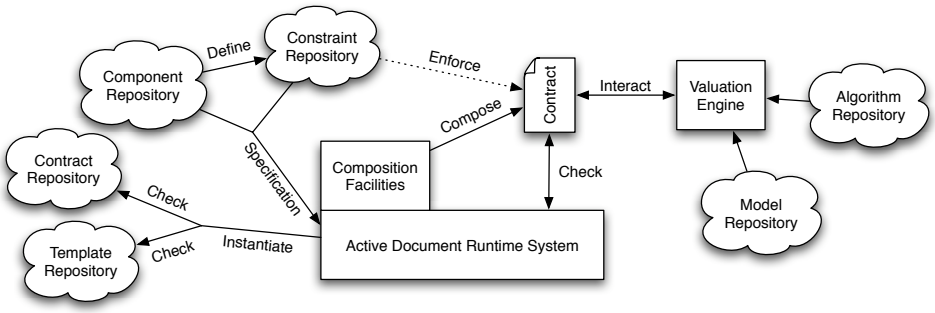

Figure 4: A conceptual overview of the software system that is developed as part of ComDeCo. *The* Active Document *runtime system acts as a kind of middleware, providing the necessary services to support composition and valuation.*

The meta-information necessary to perform this adaptation process is provided by the Active Document representing the financial contract.

Multi-dimensional tree algorithms Because the sketched usage scenario allows for derivatives based on *multiple* underlyings, computationally efficient multi-dimensional tree algorithms have to be invented. This development is based on the results of [6] which are derived from [7], providing efficient and fast converging algorithms.

4.1 Models

Modeling an european plain vanilla option by assuming the applicability of geometric brownian motion with constant volatility σ for its underlying by Black & Scholes [8] (respectively Merton [9]) in 1973 led to the first closed-form solution. Until now, ongoing research has come up with closed-form solutions for a large amount of types of derivatives (see e.g. [5]). Albeit being used as a market standard, the famous Black-Scholes model reaches its limits in case of complex derivative contracts (see [10], [11] and references therein) and is therefore only one of a plethora of models a general-purpose pricing engine has to provide and handle. In addition to the *Black-Scholes model*, at least *local* and *stochastic volatility models* will have to be integrated into the core valuation engine.

4.2 Component-Oriented Valuation

Component-oriented design principles, the foundation of the Active Document system, are dominating the valuation engine architecture, too. In contrast to the common approach of hard-coding all imaginable meaningful combinations of derivatives and models, resulting in large refactoring and reimplementation efforts in case of feature enhancement requests, the flexible generic pricing engine allows for easy extensions using components. Componentised algorithms and models are subsumed in the engine's repository. As the pricing engine is able to select the best-fitting modus operandi for fair price calculations from its repository according to the meta-information provided by the Active Document system, it is able to cope with almost arbitrary complex contracts.

Combining both, Active Document technology for easy composition of contracts and component-oriented valuation for efficient and flexible fair price calculations, provides all ingredients for the next generation tool chain for derivative contract construction.

5 Towards new horizons

Active Document technology enhances standard derivative design processes, which represent a large part of everyday work of any financial engineer. The impact of Active Documents is not limited to the workflow of ComDeCo's primary audience - the following paragraphs present a selection of applications beyond this use case.

5.1 Personalised Derivative Contracts

With derivatives being a traditional OTC product, the participating parties are formed by financially strong investors, e.g. institutional or hedgefond managers, because usual OTC is only cost-effective in case of large-scale financial investments. During the last years, the growing interest of small and medium investors for alternatives to shares and bonds has been blocked by the financial barrier formed by the high design costs of individual designed products, preventing OTC to become an option for small and medium-scale investments.

With ComDeCo, the financial barrier is lowered significantly, making OTC a realistic alternative to already available semi-individual financial products. The customer's phase of influence is expanded into the contract design process, because the customer him-

self is able to design autonomously[5]. The ACTIVE DOCUMENT system used to implement COMDECO supervises the design process by providing the set of composition operators and components for contract construction, additionally enforcing composition constraints. Effects of contract changes are instantaneous, enabling easy composition of contracts until complete conformance to individual demands is reached. Contract negotiation and design become interactive and explorative because of the user-guiding facilities of an ACTIVE DOCUMENT framework. Next generation electronic financial services will allow customers to add individually designed derivatives to their portfolios in real-time, even if the customer is not an expert in the field.

5.2 Computer-Aided Training

Using COMDECO as flexible electronic learning materials, interactive and feedback-driven training lessons help to improve knowledge about derivatives in shorter periods of time when compared to traditional training methods[6]. By interactively composing derivative contracts based on the company's component repository, employees are able to participate in learning lessons according to their individual learning habits instead of being forced to follow predetermined learning paths. In-house training lessons may substitute expensive third party offerings.

5.3 Smart Contract Repositories

After introducing a COMDECO based approach, the company's repository grows with every new contract, accumulating knowledge and therefore being a valuable asset. Before COMDECO, there were just mathematical formulae, possibly enriched with metadata, often using some kind of free-form markup. COMDECO introduce structural information above the pure implementation description based on plain formulae, making each contract a queryable entity by using specialised languages, e.g. "List all contracts having a stop loss of X and smoothing".

6 Conclusions

ACTIVE DOCUMENT technology combined with a component-oriented pricing engine provides the tools necessary to tackle the increasing design demands of current and future contract design workflows. Being highly interactive, an explorative and feedback-driven design process, both the financial engineer and the non-expert customer may profit from, becomes reality. The increased level of abstraction allows for enhanced supportive technologies that help to decrease the overall design time and reduce or even avoid potential errors or problems by shifting them towards very early stages of the design process. The traditionally strictly separated entities *Factsheet* and *Implementation* are unified by this approach, leading to increased knowledge reuse, financial engineers as well as customers may benefit from.

[5] Currently, the customer is billed (in)directly for the design process carried out by a financial engineer, making an arbitrary number of iteration steps illusory.

[6] COMDECO could provide services which are comparable to but more flexible than those of *Minerva*.

References:

[1] Box, D.: Essential COM. Addison Wesley Publishing Company Incorporated. 1999

[2] Apple Computers Inc.: Inside Macintosh: OpenDoc Programmer's Guide. Addison Wesley Publishing Company Incorporated. 1996

[3] Reitz, M. & Stenzel, C.: Minerva: A component-based framework for Active Documents. *Proceedings of the Software Composition Workshop (SC 2004)*. ENTCS 114. Elsevier. 2005

[4] EASYCOMP (IST Project 1999-14191). Easy Composition in Future Generation Component Systems. `http://www.easycomp.org`

[5] Korn, R. & Korn, E.: Option Pricing and Portfolio Optimization. AMS. Rhode Island. 2001

[6] Müller, S.: Multi-Dimensional Trees for Option Valuation. Diploma Thesis. University of Kaiserslautern. March 2006

[7] He, H.: Convergence from Discrete to Continuous-Time Contingent Claim Prices. Working Paper, University of California, Berkeley

[8] Black, F. & Scholes, M.: The pricing of options and corporate liabilities. Journal of Political Economy, **81**, 637–659. 1973

[9] Merton, R.: The theory of rational option pricing. Bell Journal of Economics and Management Science, **4**, 141–183. 1973

[10] Nögel, U. & Mikhailov, S.: Heston's stochastic volatility model. Implementation, calibration and some extensions. *WILMOTT Magazine*, 74–79. July 2003

[11] Nögel, U.: Option Pricing using Stochastic Volatility Models. *Progress in Industrial Mathematics at ECMI 2002*. Springer Verlag. Heidelberg. 2004