# COMDECO: Composable Derivative Contracts

**Introducing end-user oriented technologies to Financial Engineering**

Markus Reitz*

Software Technology Group
University of Kaiserslautern
P.O. Box 3049, 67653 Kaiserslautern
Germany
reitz@informatik.uni-kl.de

**Abstract:** Designing and valuating derivative contracts belongs to a financial engineer's daily tasks. Their almost unlimited flexibility[1] has already created the need for computer-aided techniques – typically spreadsheet-based software solutions. Lacking in-depth software development expertise promotes inflexible prototypes individually developed for each new contract to be designed and valuated. Scalability is worse and time constraints in general prevent deliberate solutions not tied to specific products. COMDECO[2] provides a conceptual and technical framework supporting efficient design and valuation methodologies for current and future contracts by adapting and augmenting state of the art software technology concepts. Based on ACTIVE DOCUMENTS, end-user oriented composition styles without the need for programming skills are supported. This paper summarises COMDECO's overall objectives, illustrates how ACTIVE DOCUMENT technology is used in financial engineering, and sketches future directions of work.

## 1 Introduction

Starting with simple *calls* and *puts* about fourty years ago, the demand for all new financial products results in a steadily growing plethora of derivatives. Albeit having evolved into an important market segment, design and valuation methodologies have only slightly changed. Paper-based termsheets specifying contracts using mathematical formulae in conjunction with textual descriptions have been superseded by spreadsheet-based solutions, but fundamental concepts remain almost the same, making current electronic representations tentative solutions. More and more, problems due to increasing complexity and the inability to easily incorporate modern concepts of software technology become apparent. Even worse, financial engineers need in-depth software development expertise,

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

[1] "With derivatives you can have almost any payoff pattern you want. If you can draw it on paper, or describe it in words, someone can design a derivative that gives you that payoff." (Fischer Black, 1995)

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

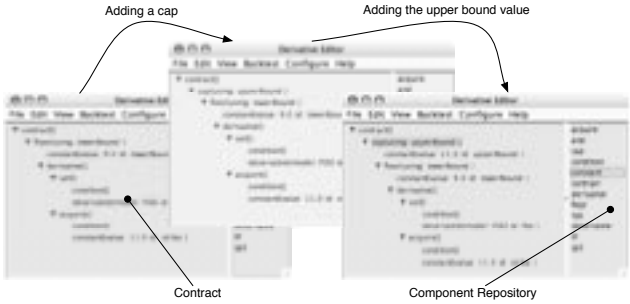


Figure 1: Visually composing a derivative contract by extending an already available one. A *cap* component is added, resulting in a *transitional document state* indicated by a yellow background. By adding an upper bound, the document reenters the valid state and is ready for valuation again.

because significant programming skills are required in order to perform necessary adaptations. In practice, financial engineers usually augment Microsoft Excel spreadsheets with functionality supplied by *dynamic link libraries* (DLL), bridged by *Visual Basic for Applications* (VBA) glue code or wrapper code. An end-user oriented tool chain letting financial engineers focus on the design and valuation of derivative contracts without the need for low-level programming is not existent.

## 2 Overview of COMDECO

One of COMDECO's main goals is a clear separation of software engineering and financial engineering aspects. A conceptual *end-user compatible framework* for the construction and valuation of derivative contracts is provided, usable without any programming skills. End-users utilise components as building blocks to create contracts. Customising and extending capabilities of the *derivative contract development environment* is performed by adding or removing components stored in a repository. The provided flexibility goes beyond that of simple preference adjustments common in current software systems.

A derivative contract is modeled as an ACTIVE DOCUMENT [Rei06], a combination of the well-known document metaphor and component-oriented principles. The resulting *hypertermsheet* resembles similarities with its fixed representation counterparts, but incorporates many advantages of current software technology. Using components as building blocks, any derivative contract is a piece of software composable by non-expert end-users. Creating a derivative contract means selecting appropriate components from the system's component repository and combining them by intuitive drag and drop gestures (see Figure 1). Technically, composition is based on the *decorator pattern* [GHJV97] allowing for an iterative contract construction by composing appropriate components. Besides components as fine-grained building blocks, already designed contracts stored in repositories act as coarse-grain units a financial engineer can make use of. Repositories grow with every newly designed contract, making them an important value asset. For long-term persistence,

an ACTIVE DOCUMENT modeling a financial contract is rendered to a XML-based representation. Besides simplifying interaction with third party tools, this representation is an alternative to visual composition techniques, too. Being a high-level declarative description[3], *textual composition* is easier to handle for end-users, because of a domain specific vocabulary removing the impedance mismatch which is present in case of general-purpose programming language usage[4]. The provided vocabulary largely maps to components directly, e.g. Figure 1 demonstrates a composition with *contract*, *floor*, *cap*, *constant*, *observable*, *derivative*, *sell*, *acquire*, and *condition* as involved components[5].

Operating on high-level concepts, e.g. *caps* and *floors*, instead of possibly ambiguous mathematical operators such as $\min$ or $\max$ provides additional advantages far beyond improved readability and optimised training curves. Structural constraints such as "Any *cap* component has to be related to either a *observable* or a *constant* component." and semantical constraints such as "For the loss protection barrier $F$ and the profit limitation $C$ applied to an underlying's performance, the condition $F < C$ should hold."[6] are checkable by the ACTIVE DOCUMENT runtime system, guiding the user during the composition phase, possibly providing hints supporting an *explorative composition style*.

In order to trade contracts, the fair price has to be determined. At least three categories of methods have to be considered: *closed-form solutions*, *Monte Carlo simulation*, and *tree-based algorithms*. Although tree-based algorithms represent financial engineering's swiss army knife, using closed-form solutions when possible significantly reduces computationally efforts. Under certain circumstances, Monte Carlo simulation might be desirable, too. Moreover, valuation models such as *Black & Scholes* or *local & stochastic volatility models* have to be taken into account, further complicating the valuation phase of derivative contracts. Hence, a categorisation scheme making use of additional meta-information gained by querying the ACTIVE DOCUMENT representing the derivative contract to be valuated allows for (ideally) automatic selections of best-fitting approaches. Instead of manual programming efforts, appropriate algorithms & models are automatically selected, performing fair price calculations with little or no user interaction, removing currently common error-prone programming tasks without loosing flexibility. As in the case of vocabulary extensions, available valuation capabilities can be extended by deploying supplemental components. In contrast to common isolated applications, deployment affects the whole system, i.e. valuation *and* design facilities, triggering automatic adaptation of the system to the new configuration.

An end-user oriented approach should be seamlessly widenable to a larger scope of users without requiring changes to the conceptual model. The following scenario would be a straightforward extension when web-enabling COMDECO's tool chain.

The interest of small and medium investors in alternatives to stocks and bonds has been rising for years, but individually tailored products do not yet exist. Derivative contracts are

[3]The suitability of functional programming languages such as Haskell for the description of financial contracts was pointed out in [JE05].

[4]In fact, COMDECO's XML description is a *domain-specific language*.

[5]For a detailed discussion concerning these and other aspects refer to [RN06].

[6]In case of $F \geq C$, the payoff function characterising the contract ($P = \min(\max(X, F), C)$) would be independent of X, which is almost always not the intended design.

OTC[7] products only offered to large scale investors due to high break-even points. Additionally, small and medium investors cannot be expected to have the necessary know-how to conduct negotiations autonomously, making the financial engineer carrying out design and valuation for each contract an important expense factor. Besides not requiring programming expertise, COMDECO provides a high-level description language for derivative contracts which operates on concepts, not formulae, requiring less mathematical knowledge. Investors will be able to construct contracts by themselves, using a specific set of components provided by next generation online banking systems. The result will be checked & valuated and acceptance or rejection of the offered conditions is decided just in time by the investor. The transaction is completed by charging the investor's account with the appropriate amount of monetary units in case of acceptance.

## 3 Conclusions

Financial Engineers are usually not software engineers, creating the need for *end-user compatible frameworks* which do not require programming skills to design and valuate derivative contracts. COMDECO provides such a framework based on ACTIVE DOCUMENT technology. Derivative contracts are modeled as ACTIVE DOCUMENTS, combining state of the art computer science concepts and an intuitive document metaphor, resulting in a *hyperdocument*. By making use of end-user oriented technologies, the potential audience of COMDECO's derivative contract design tool chain goes beyond financial engineers. Next generation online banking systems, allowing a *person in the street* (PITS) [Ras04] to autonomously design derivatives using an explorative and guided composition style are possible application scenarios for the concepts and techniques provided by COMDECO.

## References


[GHJV97] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1997.

[JE05] S.L. Peyton Jones and J.-M. Eber. *How to write a financial contract*, volume Fun Of Programming of *Cornerstones of Computing*. Palgrave Macmillan, 2005.

[Ras04] J. Raskin. *The humane interface: new directions for designing interactive systems*. Addison-Wesley Pearson Education, 2004.

[Rei06] M. Reitz. Active Documents - Taking advantage of component-orientation beyond pure reuse. In *Proceedings of the 11th Workshop on Component-Oriented Programming (WCOP)*, 2006.

[RN06] M. Reitz and U. Nögel. Derivative Contracts as Active Documents - Component-Orientation meets Financial Modeling. In *Proceedings of the 7th WSEAS International Conference on Mathematics and Computers in Business and Economics (MCBE)*, 2006.


[7] Over The Counter, i.e. terms and conditions are negotiated between vendor and vendee in contrast to off-the-shelf financial products.