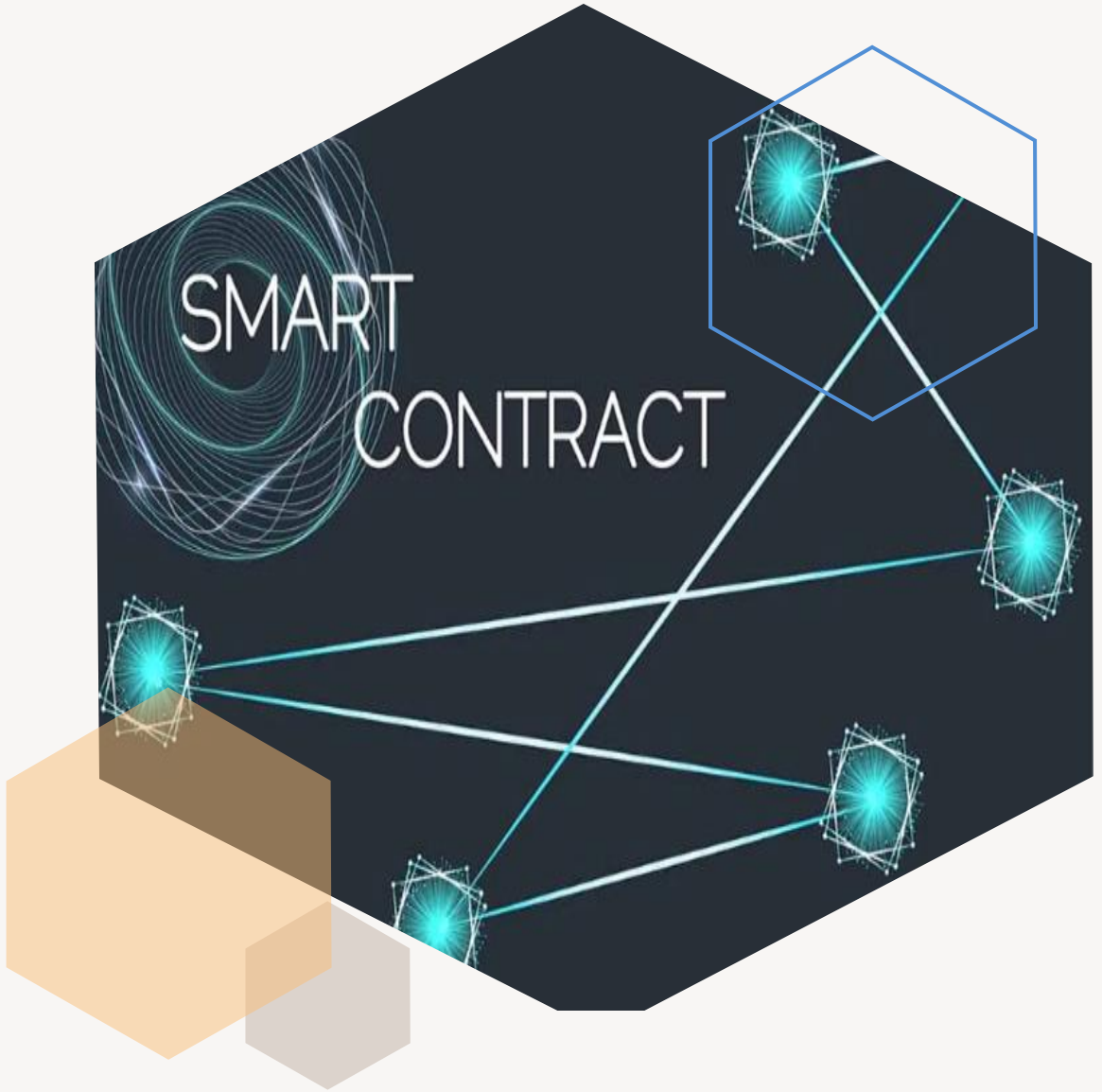
# Smart Contract Templates



# Agenda


What Are Smart Contract Templates?
Who Proposed This Idea?
Development of Smart Contract Languages
Domain-Specific Languages (DSLs) for Finance
How DSLs Help in Smart Contracts
The Future
Conclusion

# What Are Smart Contract Templates?

**Smart contracts are self-executing agreements written in code.**


**smart contract templates**

predefined contract structures that provide a legal and technical framework for financial agreements.

## WHO PROPOSED THIS IDEA?

In 2016, **Clack et al.** introduced the concept of smart contract templates.

Their paper titled 'Smart Contract Templates: **Foundations, Design Landscape, and Research Directions**' laid the groundwork for this innovation.

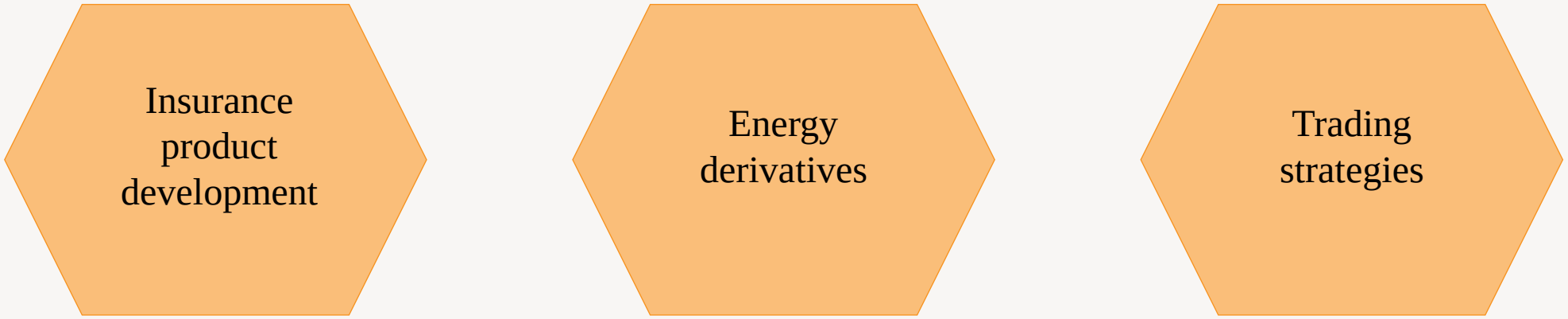
# Development of Smart Contract Languages

To support smart contract templates, researchers are working on a new language called **CLACK (Common Language for Augmented Contract Knowledge).**
This language aims to:
•Integrate legal text with smart contracts.
•Be compatible with multiple platforms.
•Support cryptographic functions for security.

Another paper by Clack et al., **"Smart Contract Templates: Essential Requirements and Design Options"**

# Domain-Specific Languages (DSLs) for Finance

Contracts in finance have been around for a long time, and special programming languages, known as **Domain-Specific Languages (DSLs),** have been developed to handle financial transactions. These DSLs are optimized for specific financial applications, such as:



# How DSLs Help in Smart Contracts

Unlike general-purpose languages (like Java or Python), **DSLs are designed for a specific use case.** Some examples in blockchain development include:

**The DSLs used in following purposes:**

- Safety and Security
- Easier Contract Development
- Gas Optimization
- Formal Verification

## ➢ Safety and Security:

- Blockchain transactions are irreversible, so security is crucial.
- DSLs include built-in security checks to **prevent hacks and errors.**
- Example: **Solidity** helps avoid common attack vectors like **reentrancy attacks.**

## ➢ Easier Contract Development:

- DSLs abstract complex blockchain interactions, making it easier to write and understand smart contracts.
- So, it will **reduce the unnecessary complexity** of contract.
- Example: **Clarity** (used in Stacks) avoids unexpected behavior by removing unintended side effects.

## ➢ Gas Optimization:

- Execution on a blockchain requires "gas" (computational cost). DSLs optimize code execution to **reduce unnecessary operations and save costs.**
- Example: **Vyper** (Ethereum alternative to Solidity) is designed to be more gas-efficient.

## ➢ Formal Verification:

- Some DSLs allow **mathematical proof** of contract correctness.
- This reduces **bugs and unexpected behavior.**
- Example: **Michelson (Tezos)** supports formal verification.

# The Future: Graphical Smart Contract Development

A future possibility is **graphical domain-specific languages**, which allow **non-programmers** (like lawyers or financial experts) to create smart contracts using a drag-and-drop interface. The idea is similar to **Tibco StreamBase**, which enables non-developers to build trading systems.

This approach would let users visually design smart contracts, test them, and deploy them on a blockchain—all without writing code.

# Conclusion

- **Smart contract templates** aim to simplify legal and financial agreements.
- **New languages (like CLACK) are being developed** to support smart contract automation.
- **DSLs are already used in finance** and can be adapted for smart contracts.
- **Graphical interfaces could allow non-programmers** to create and deploy smart contracts easily.

# Thank you