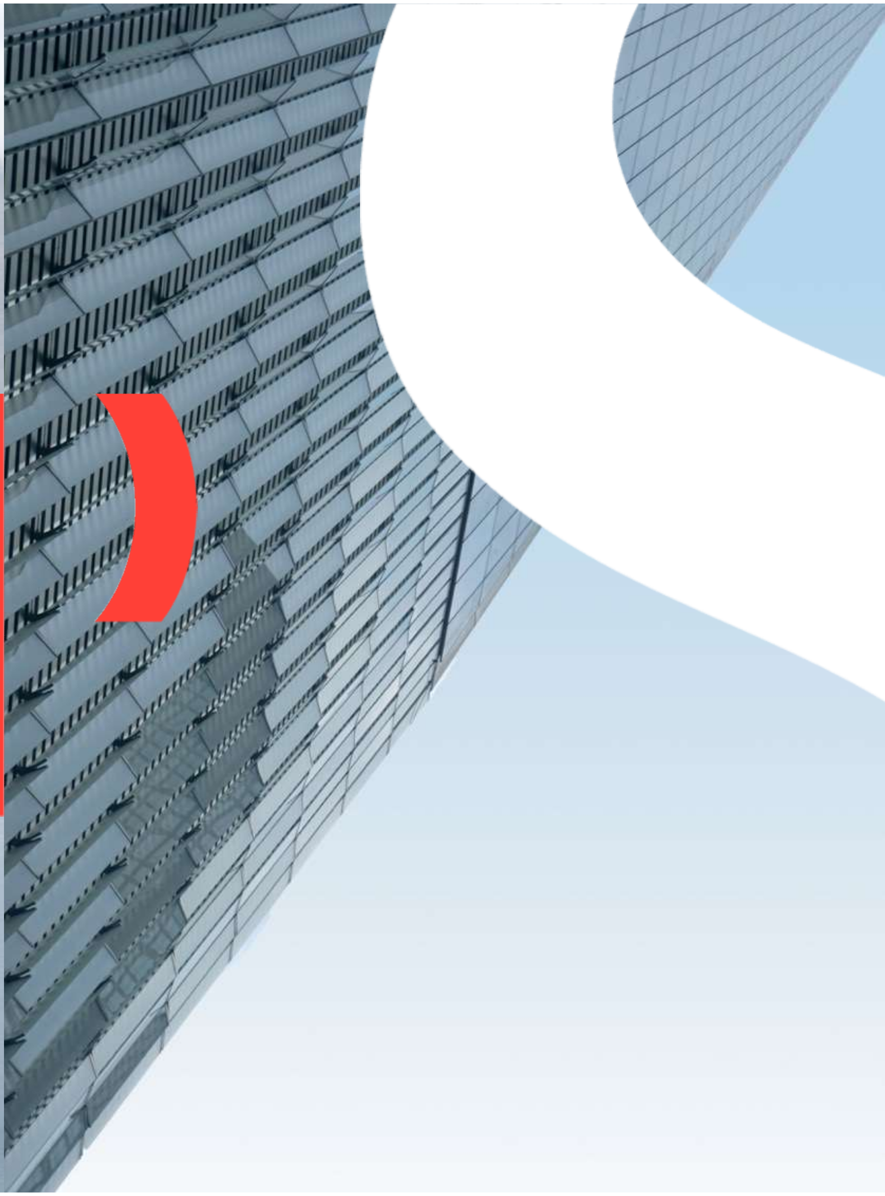
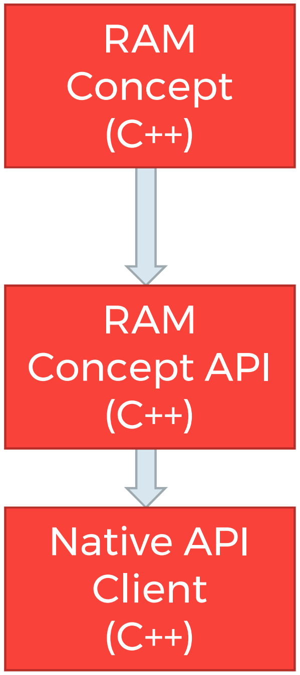


RAM

Tom Mendez ,
Senior Engineer

First Pass



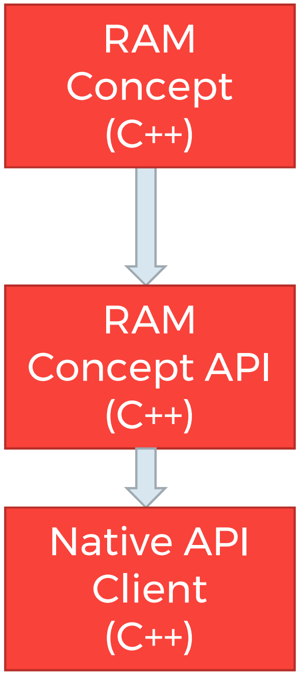
RAM Concept (C++)

RAM Concept API (C++)

Native API Client (C++)

- RAM Concept
  - *Written in C++*
- RAM Concept API
  - *Written in C++*
  - *Compiler and OS Independent*
  - *Binary Compatible*
    - Pure Abstract Interfaces
    - C Style Naming without Templates
    - No Memory Allocation Border Crossings
  - *Deliverables*
    - Library (.dll)
    - Headers (.h)

First Pass


RAM Concept (C++)

RAM Concept API (C++)

Native API Client (C++)

- Pros
  - *API code can directly reference existing C++ code*
  - *New API versions can be released without breaking client code*

- Cons
  - *API client access from languages other than C++ is difficult*

Evaluation of Needs (and Wants)

- API must be accessible from different languages
  - *VBA*
  - *.NET (C#, VB)*
  - *C++*

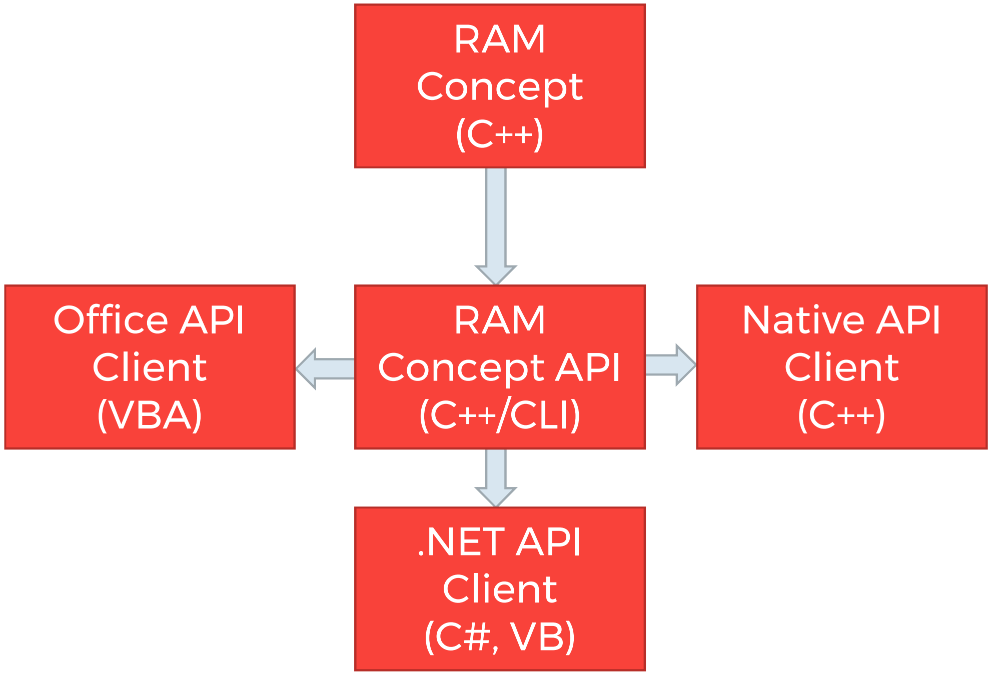
- API will likely only ever be used on a windows machine and libraries will be built with MSVC

- API should be object oriented
  - *Objects*
  - *Properties*
  - *Methods*
  - *Events*

Implementation

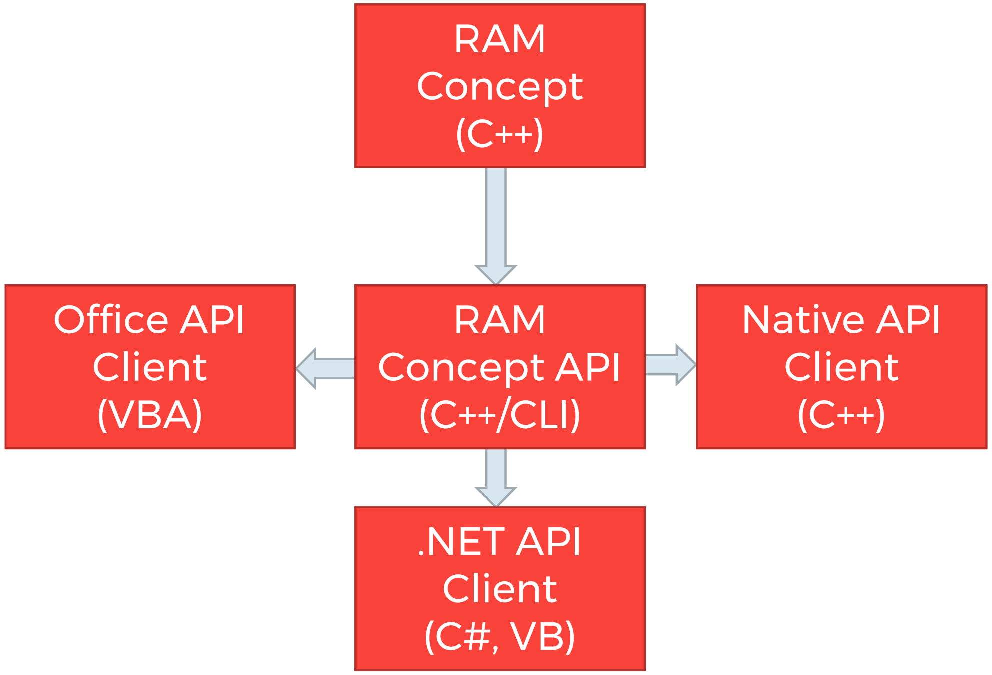
- API must be developed with COM support in order to maintain binary compatibility between languages
- API should be developed with IDispatch (OLE Automation) support to expose an object oriented set of members
- API development options
  - *C++ with COM support via ATL and MFC*
  - *.NET with COM support*

Second Pass



- RAM Concept
  - *Written in C++*
- RAM Concept API
  - *Written in C++/CLI*
  - *Exposes IDispatch COM Interfaces*
  - *Compiled for Windows with MSVC*
  - *Deliverables*
    - Library (.dll)
    - Type Library (.tlb)

Second Pass



- Pros
  - *API code can directly reference existing C++ code*
  - *New API versions can be released without breaking client code*
  - *Any COM or .NET aware client can access the API*
  - *Elegant error handling*