Une Grande École sans frontières

A Grande Ecole without boundaries

École Centrale Paris

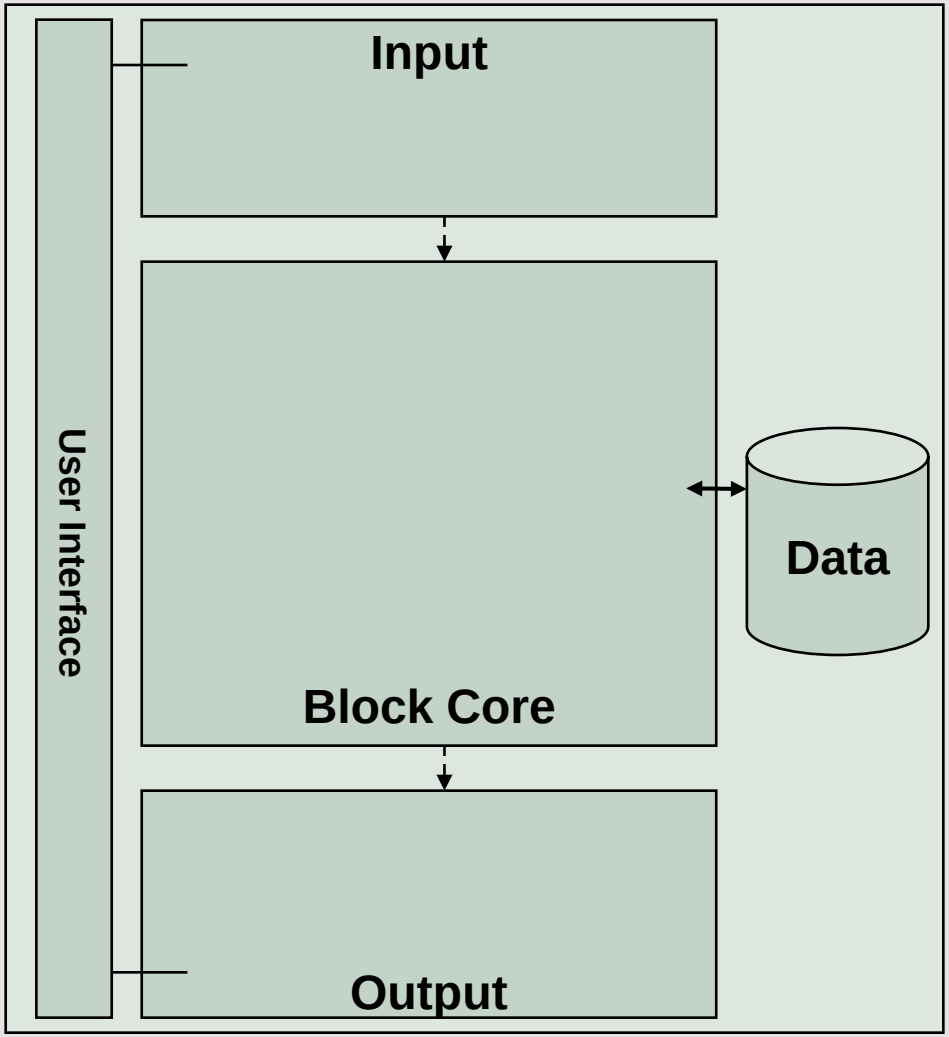
CEISAR

Software Structure

Main Parts of a Block

Let's consider a Block as a stand-alone piece of software.
We will describe later on how different Blocks communicate.

The main part of a Block is called « **Block core** »: it includes main software parts.
The Objective of most Blocks is to produce **Outputs.**
To produce these Outputs, Block Core requires to access **Data** and to be fed by **Input**.
User Interface helps user to manually enter Inputs or obtain Outputs.



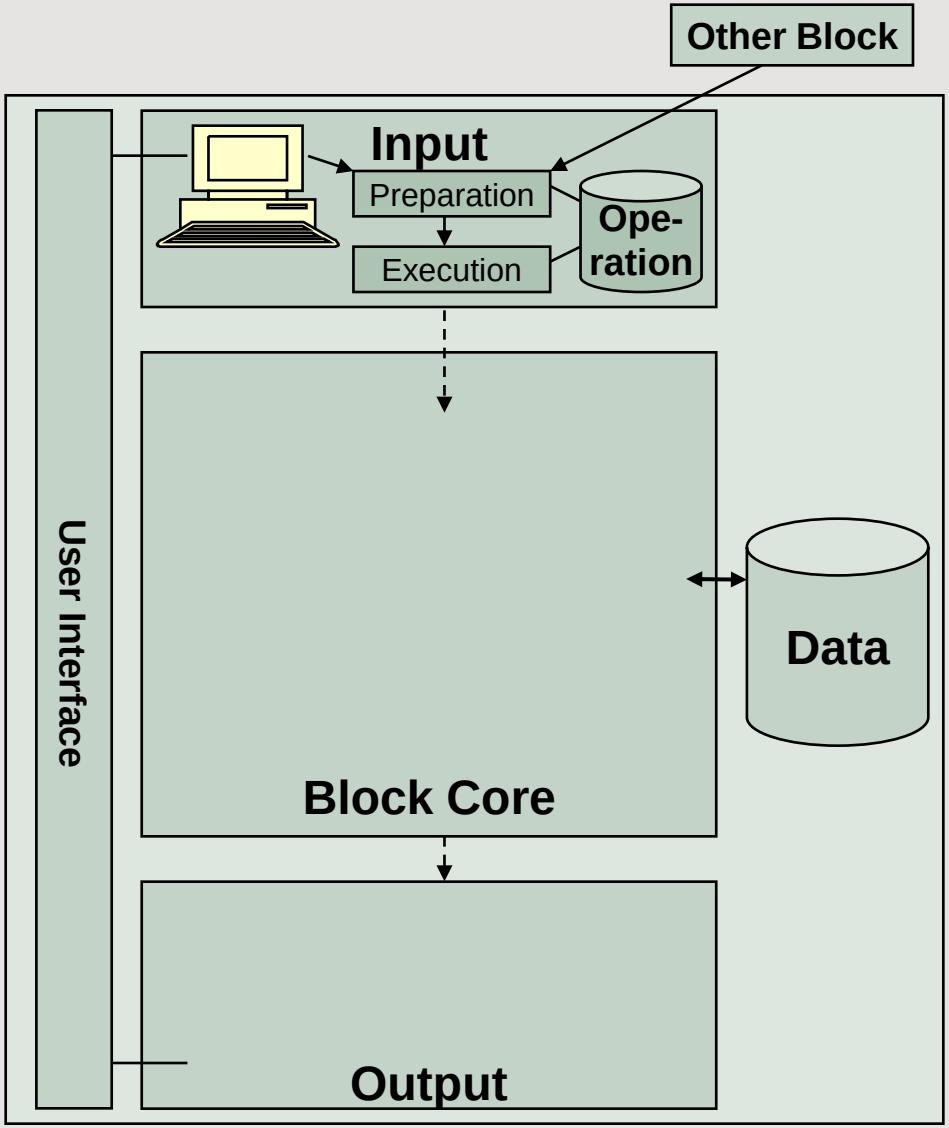
Inputs of a Block

The Input can be **Human** Input or **Automatic** Input coming from another Block.

Input can be **Synchronous** or **Asynchronous** (which means « rejects » and more complexity)

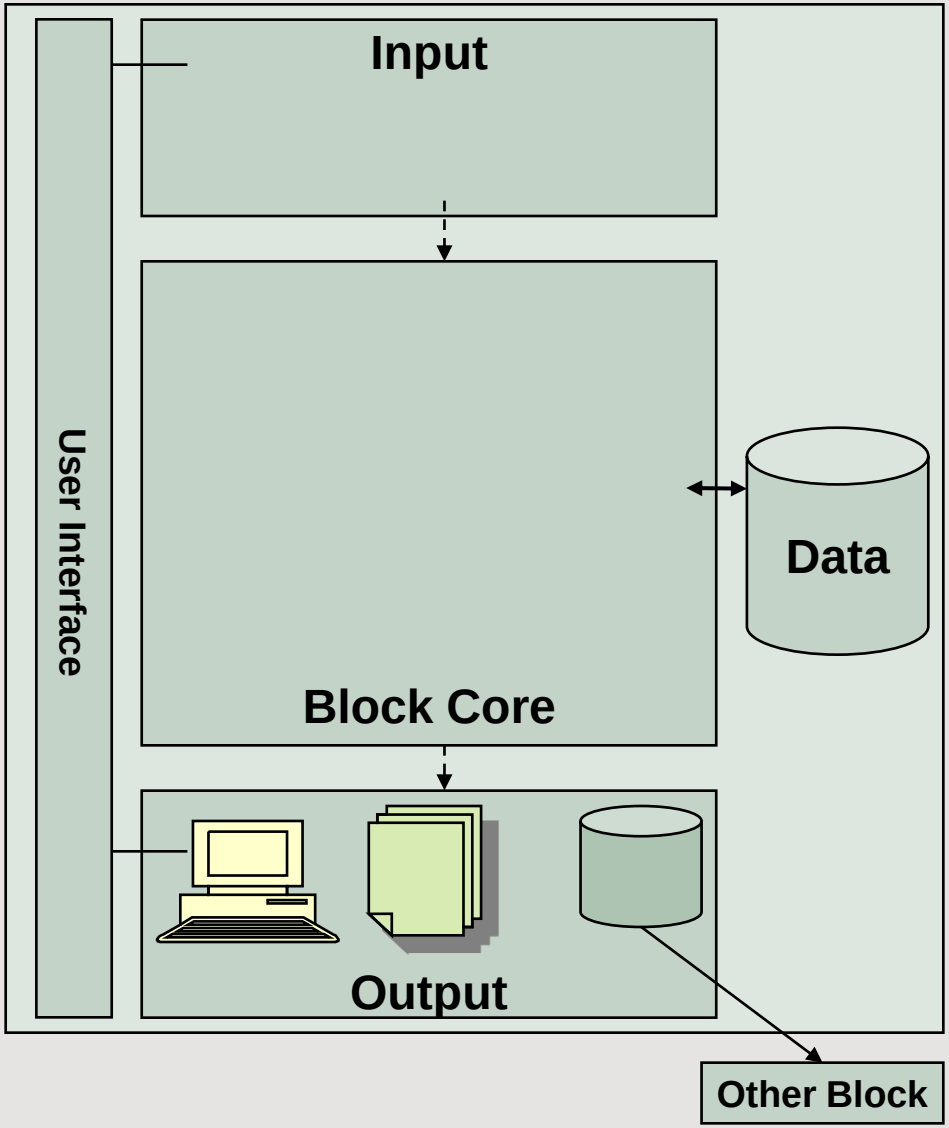
Input is decomposed into 2 phases:
Preparation (one or several Tasks)
- Data Controls
- Authorization

+ Execution when Complete, Controlled, Scheduled and Authorized
- Irreversible actions like **data updates** or **outputs** for external Systems



Outputs of a Block

Human output: inquiries or printings
Automatic output: files which become **input** for other Blocks



Block Core

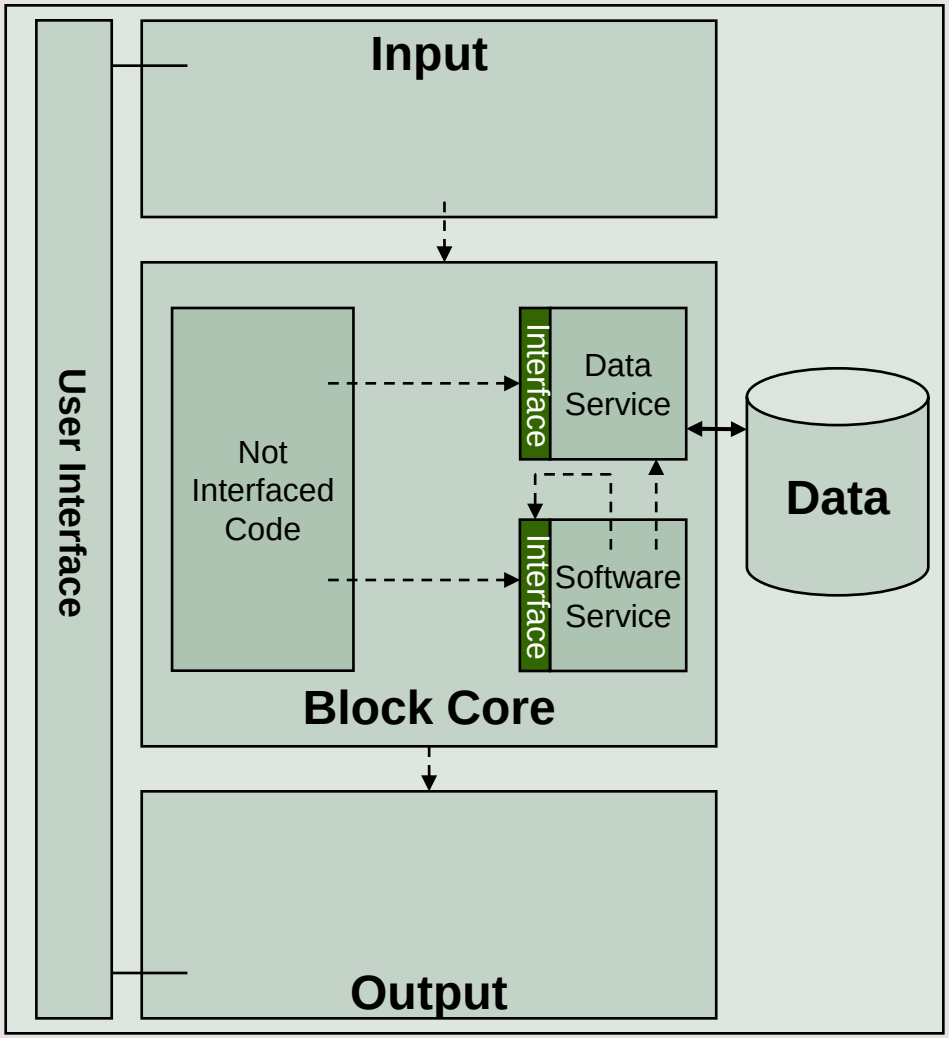
Block Core is built with
- **Not interfaced Code**
- Interfaced code: called « **Services** ». Services are the Smallest Software **Components**.

A Block is high quality if “Not Interfaced Code” is small compared to Services.

A Service =
- **Interface** (how to call it)
- **Implementation** (what it does)

Access to data must be done through **Data Services** to protect caller from data modifications.

Other services are called **Software Services**.
Software services and “Not Interfaced Code” may call Software Services and Data Services.
Data Services do not call Services.

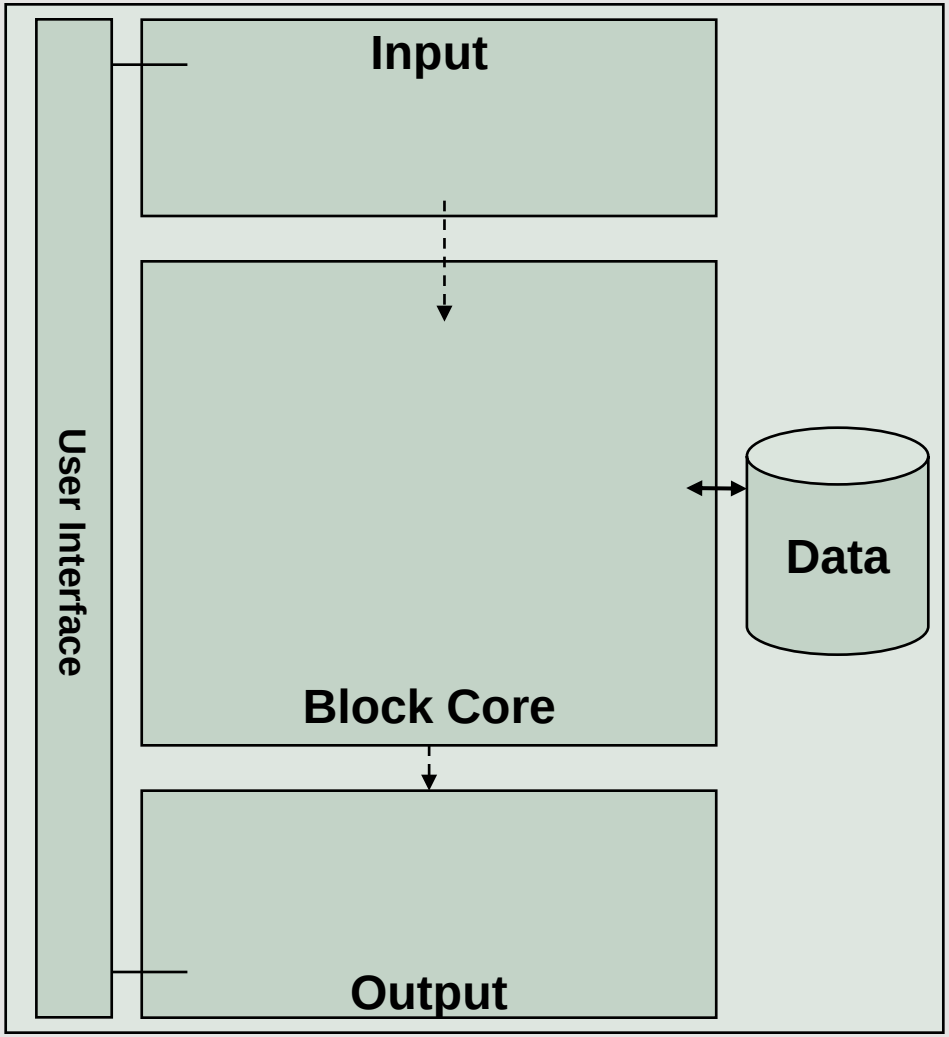


User Interface

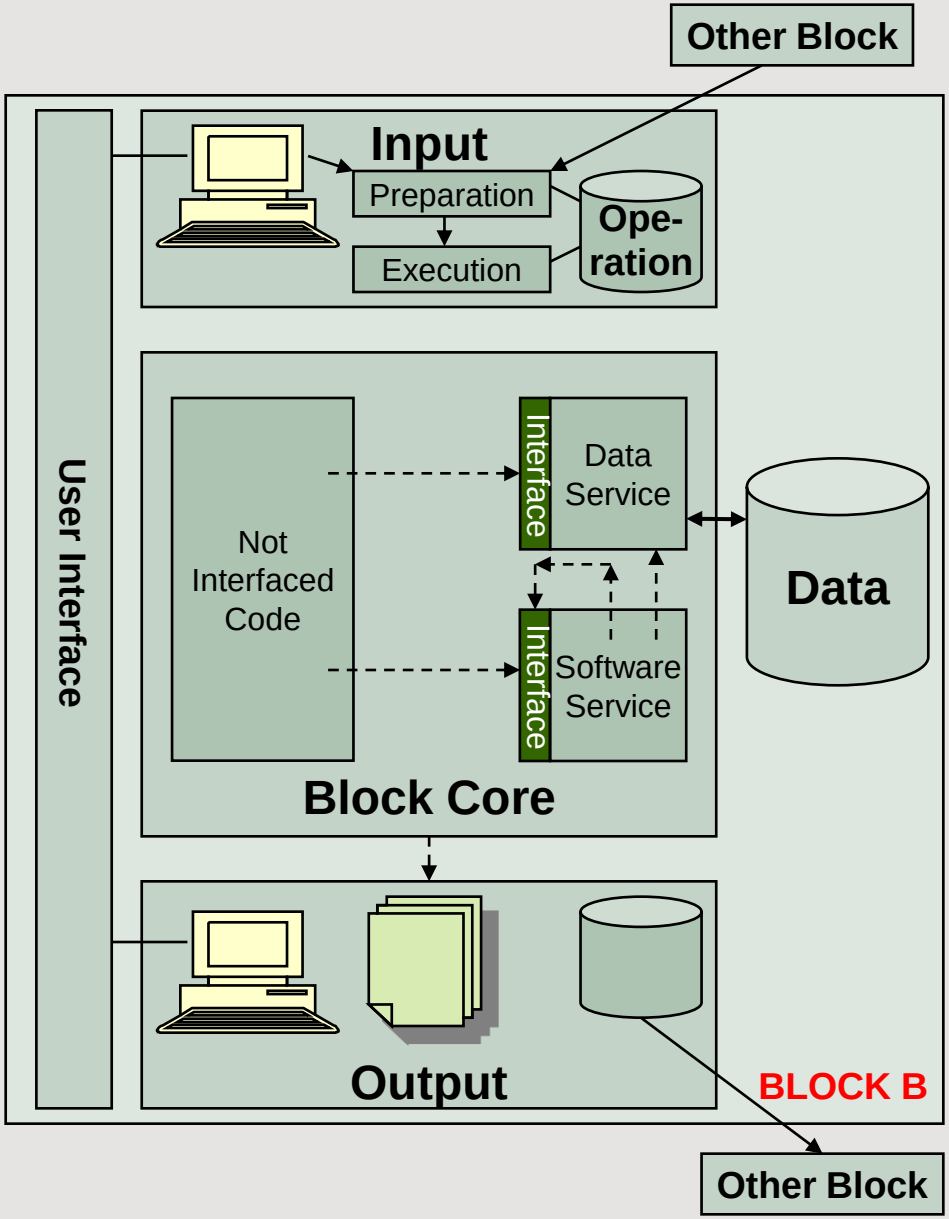
Internal or **external** Users access the Block through User Interface to Input data or obtain data from the Block.

User Interface Software includes
•Navigation
•Presentation

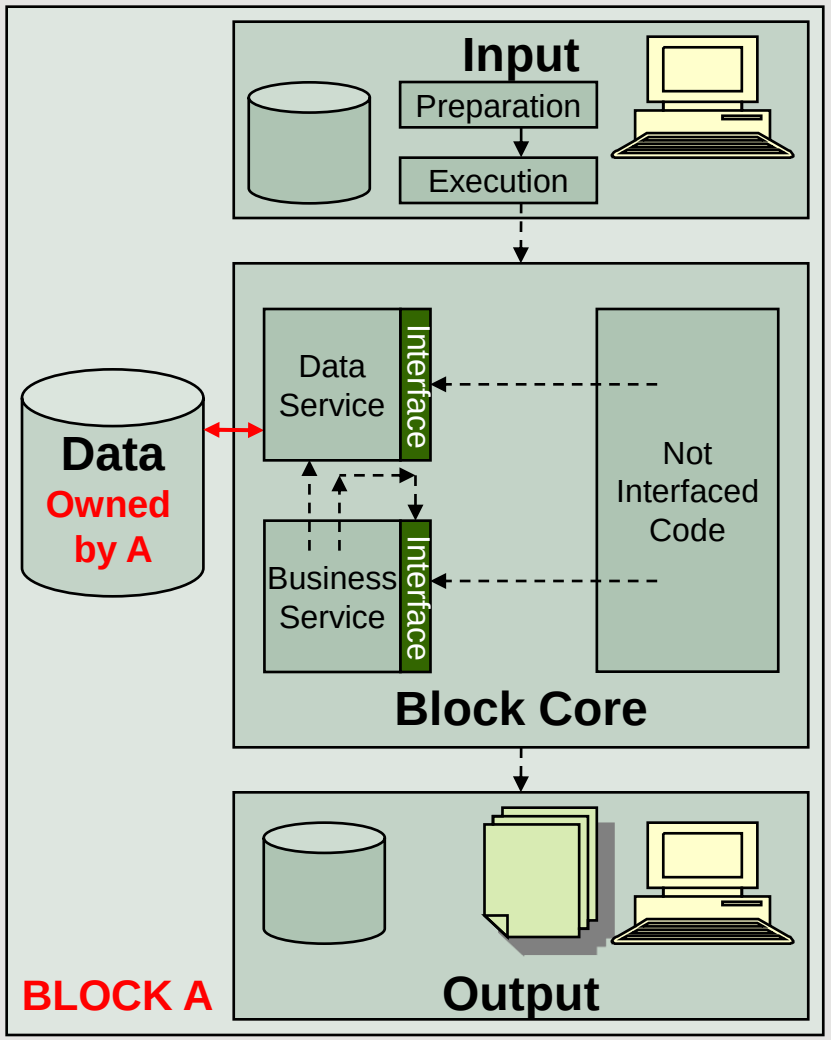
Rich or light User Interfaces can be used, but they should all **reuse** same Data services, same Software Services and same Execution mechanisms for Input.

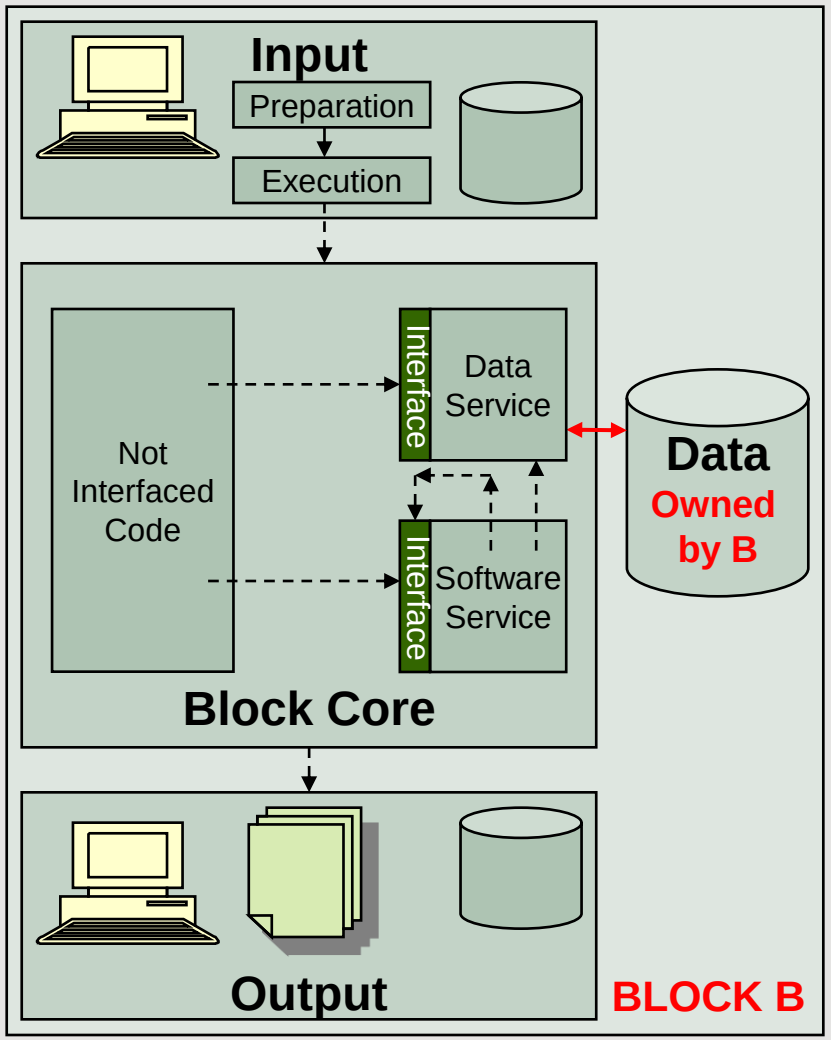


A Block


Other Block
Input
Preparation
Execution
Ope-
ration
User Interface
Not
Interfaced
Code
Interface
Data
Service
Interface
Software
Service
Data
Block Core
Output
BLOCK B
Other Block

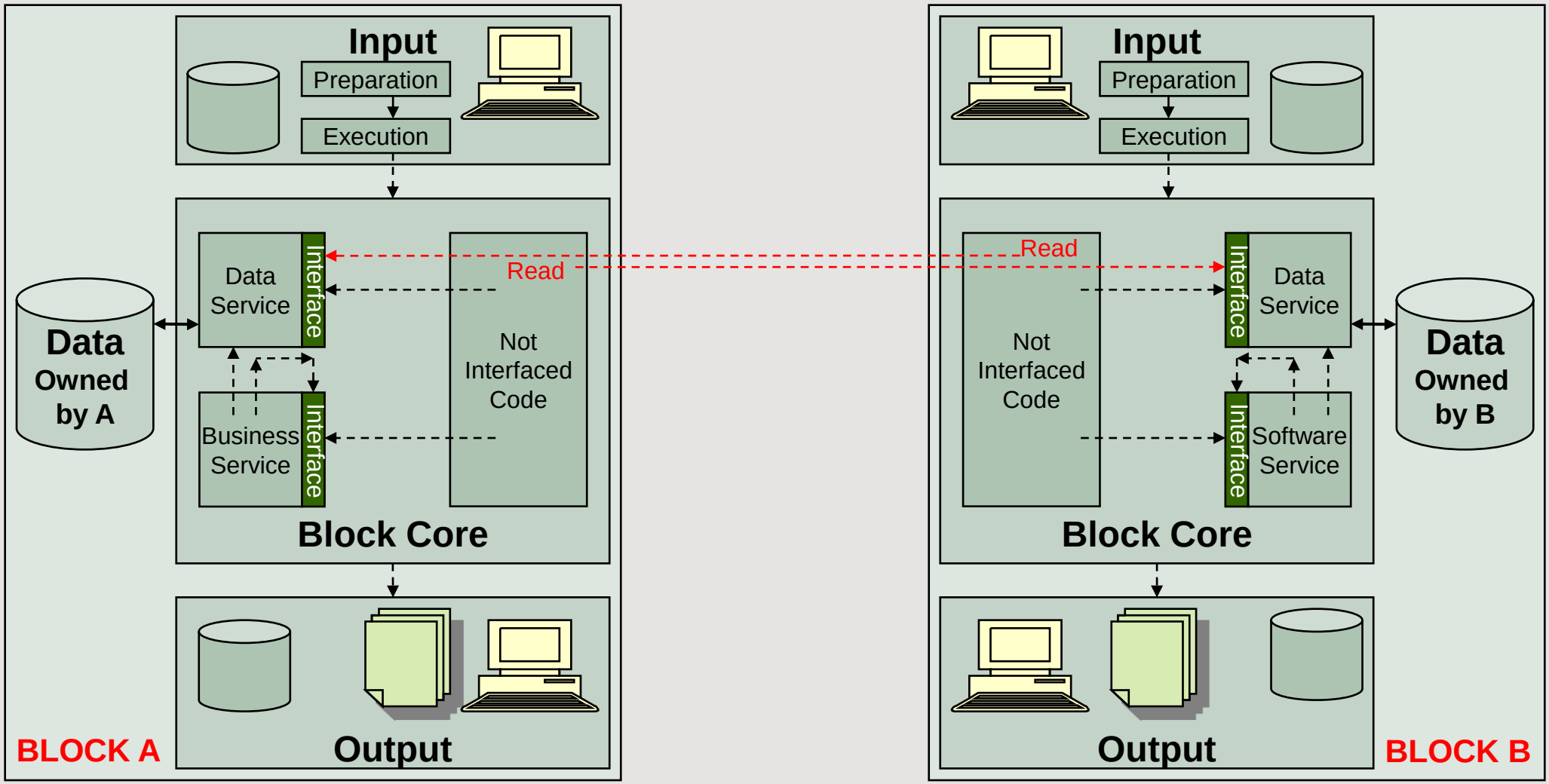
A Data only belongs to one Block





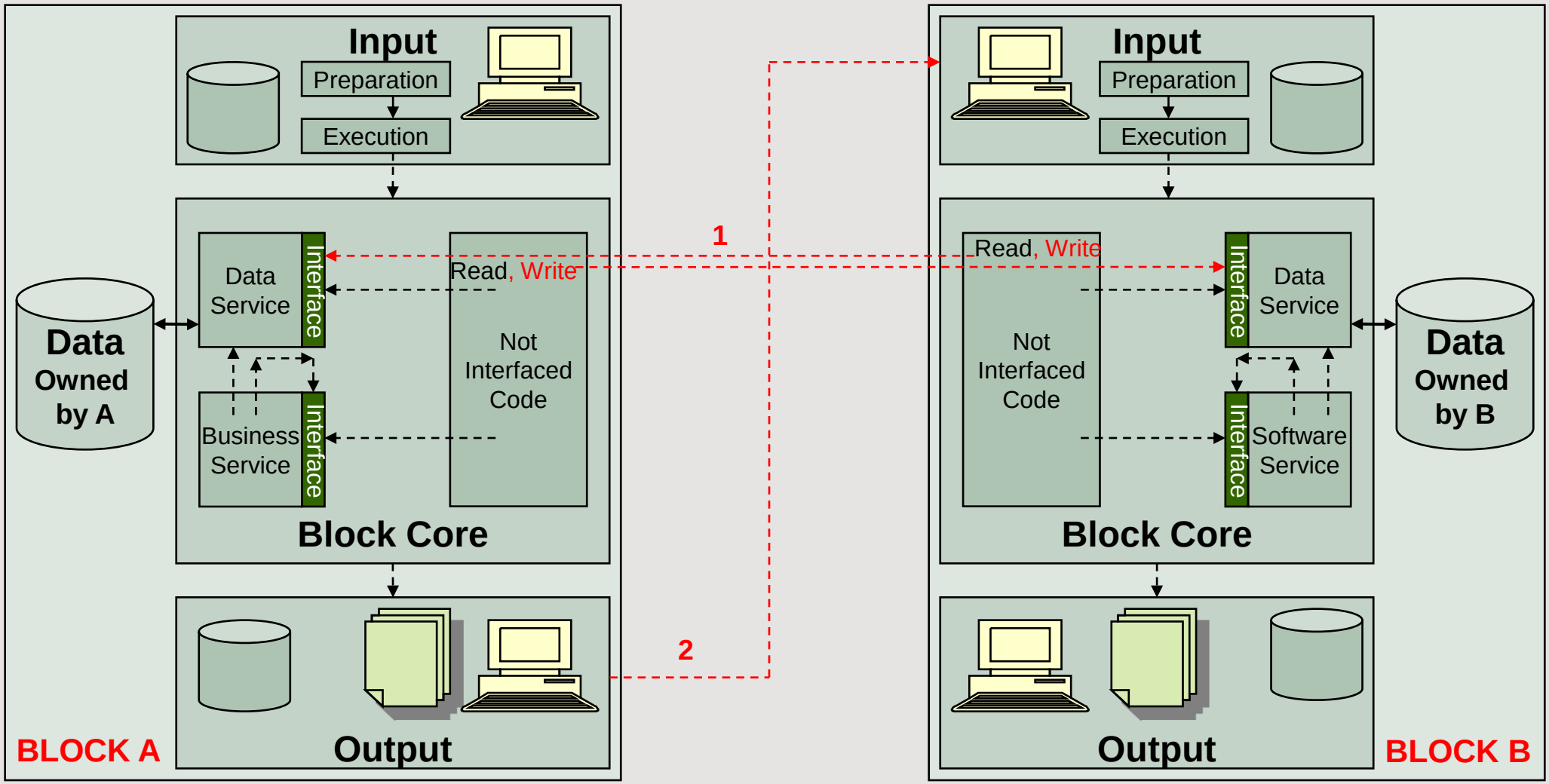
If not:
Duplication of data entries and **Inconsistencies**
For example: how to manage a common **Customer** ?

Read « External data » through Data interfaces



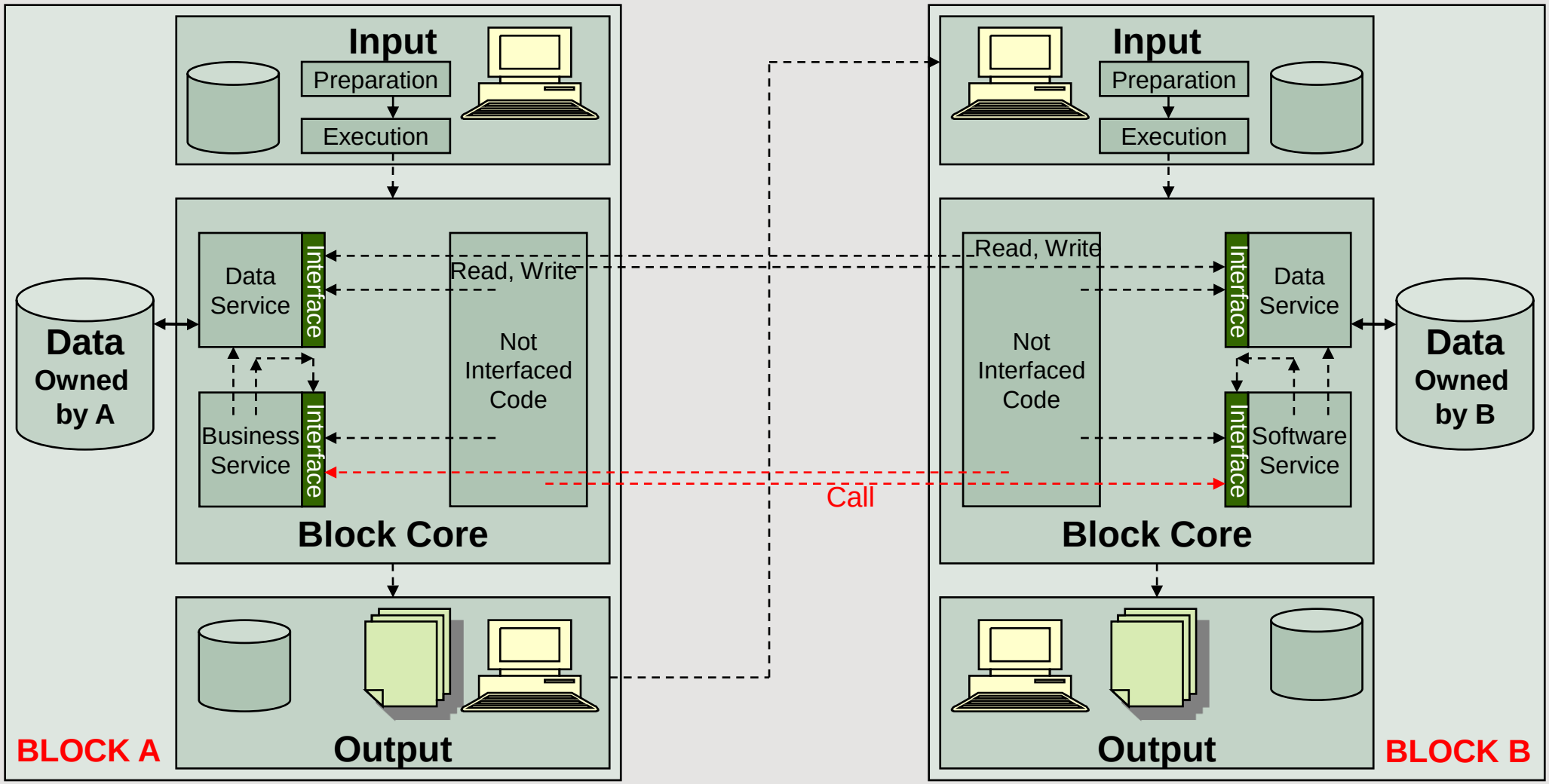
Or **Data Replication**: subscription mechanisms

Write « External Data » through Input or Write Interfaces

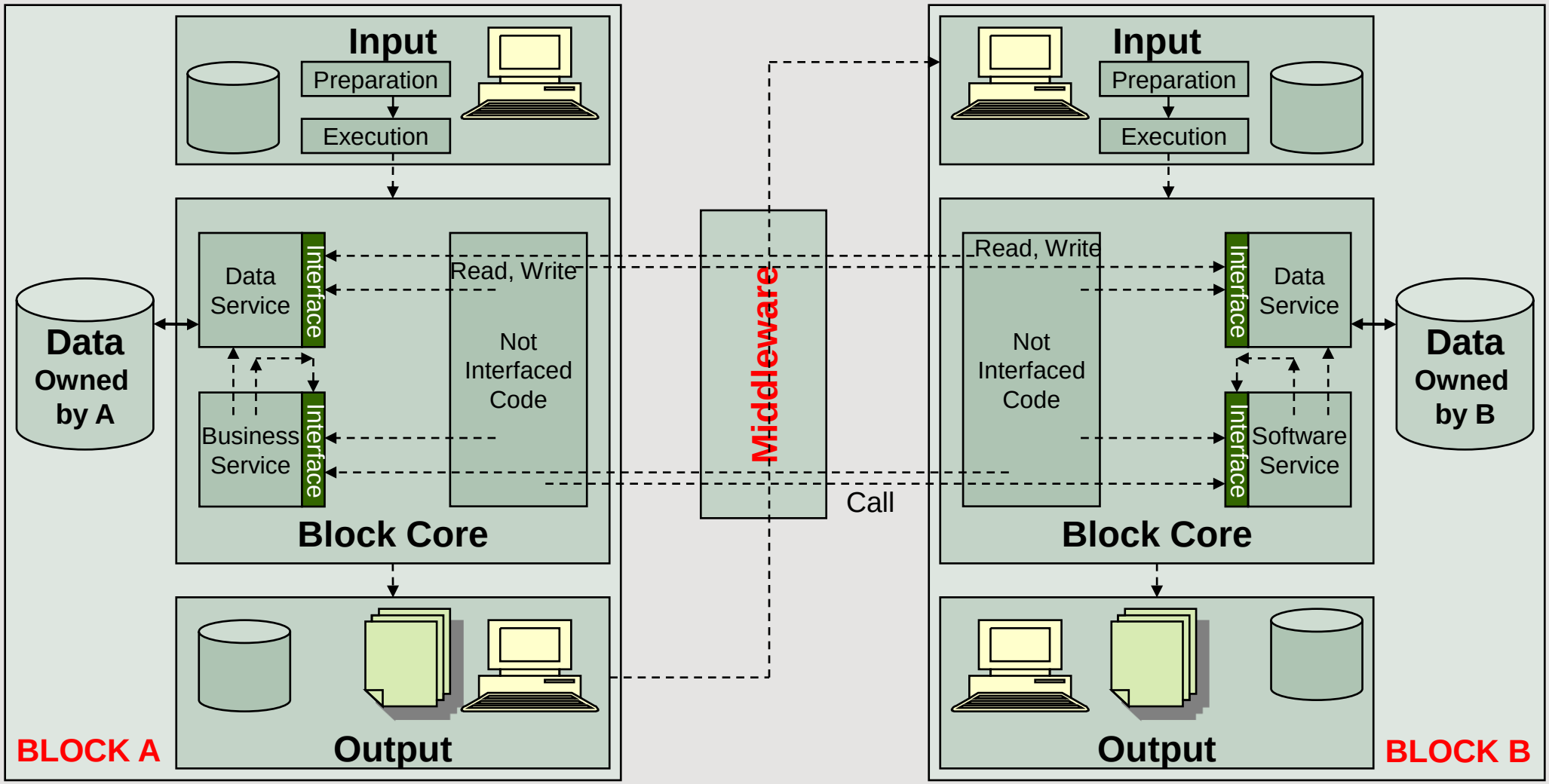


Direct Write looks **simpler** for developer and provides good **performance**
But use of **Input** solves: **journal**, **asynchronous** execution, **reuse** of input controls and authorization, **transaction**, ...

Call Services from other Blocks through Interfaces



Middleware

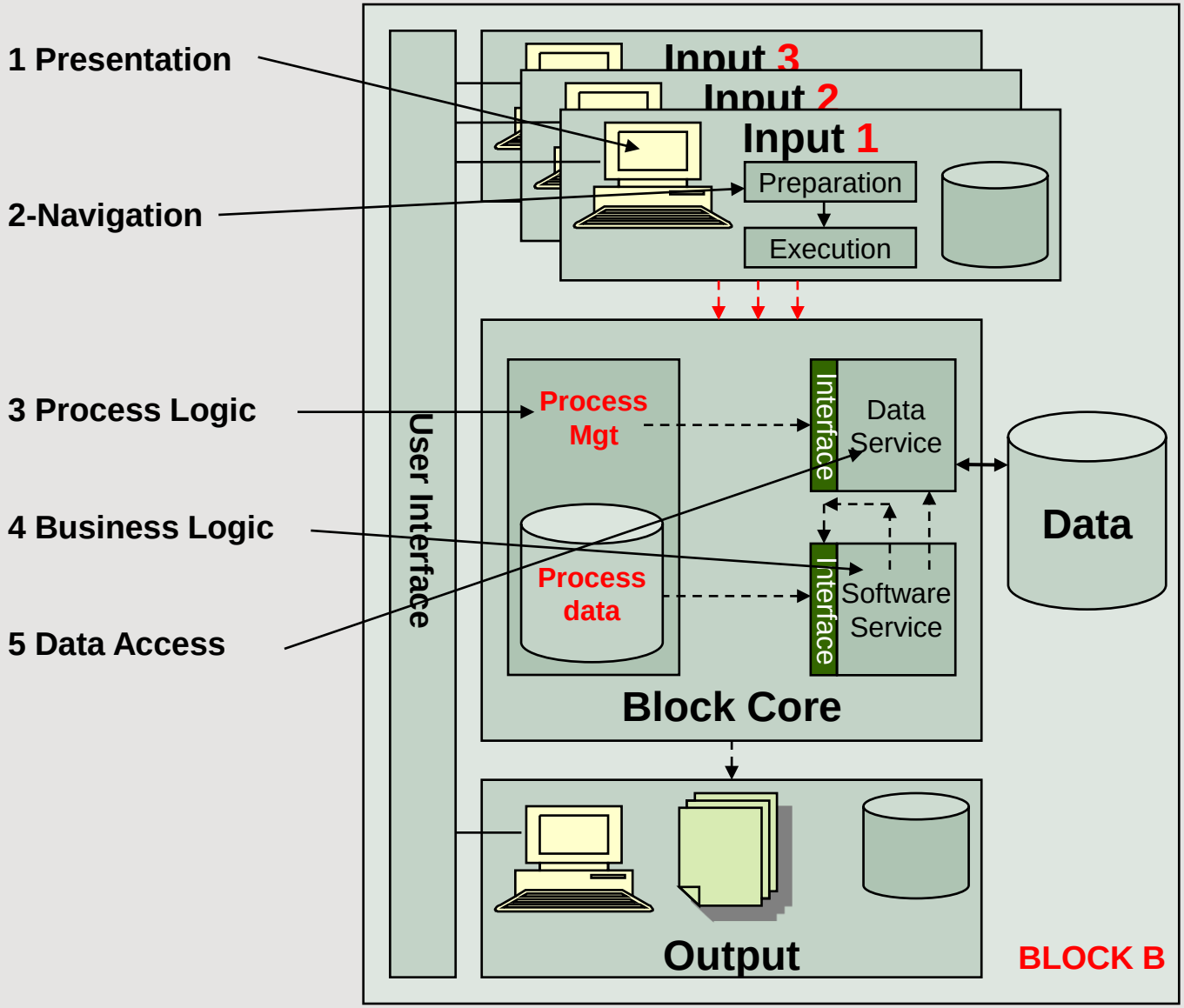

Input
Preparation
Execution
Data Service
Interface
Read, Write
Not Interfaced Code
Business Service
Interface
Block Core
Output
Data Owned by A
BLOCK A
Middleware
Call
Input
Preparation
Execution
Read, Write
Interface
Data Service
Not Interfaced Code
Interface
Software Service
Block Core
Output
Data Owned by B
BLOCK B

5 Tiers Block Structure

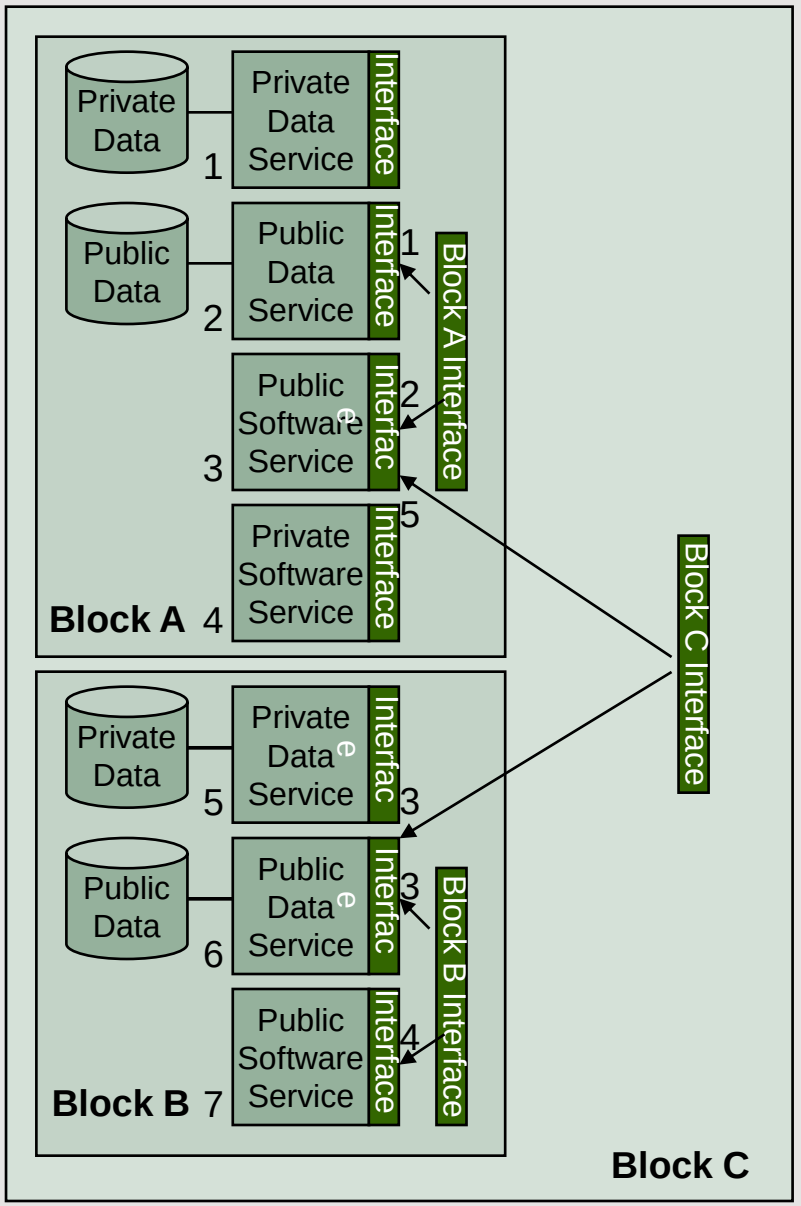
A **simple Process** only requires one execution.
Ex: Address change

A **complex Process** requires several successive executions.
Ex: claim process or Loan contract process.

A complex Process is managed by one Block, but may call Services from other Blocks, or generate input to other blocks.



Embedded Blocks: Private and Public services



To be able to create, modify, improve Blocks, they must be as **isolated** as possible.
This is why we split Services into 2 categories:

- **Private** Services are only usable from inside the Block
- **Public** Services are usable from inside and outside the Block.

Interface of Block A only contains interfaces of its Public Services (2 and 3).
Data only accessible by Private service are ignored from outside, and called Private Data.

This is a **recursive** definition. If a Block C contains Block A and B, its interface is only composed of Services accessible from outside Block C (3 and 6 in our example): the Interface of Block C is a subset of the interfaces of its embedded Blocks.

External Blocks which do not belong to the Enterprise System are only known by their interfaces.