|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | |  | | **제10회 공개SW 개발자 대회 결과 보고서** | |  | |

**□ 참가팀 개요**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **구 분** | | **세부내용** | | | |
| **팀 명** | | 삼촌 | | | **총 인원 (** 5 **명)** |
| **팀 구 성** | | **성명** | **소속** | **부서/학과** | **직위/학년** |
| **팀 장** | | 남정호 | 히즈윌 | 부설연구소 | 사원 |
| **팀 원** | | 이정훈 | 성균관대학교 | 휴먼ICT융합학과 | 석사과정 |
| **팀 원** | | 이다빈 | LeeSet | 개발팀 | CEO |
| **팀 원** | | 이두영 | LeeSet | 개발팀 | CEO |
| **팀 원** | | 이아름 | 한양대학교 | SOC 연구실 | 석사과정 |
| **참가**  **지원** | **부문** |  | |  | |
| **분야** |  |  |  |  |
|  |  |  |  |
| **지정과제**  **(참여시 선택)** |  | | | |
|  | | | |
|  | | | |
|  | | | |
|  | |  | | | |

**□ 참가팀 활동 사진**



**□ 출품작 결과 보고서**

|  |  |
| --- | --- |
| **출품작명** | 삼촌 프레임워크 |
| **프로그램 등록URL** | <https://github.com/samchon/framework>  <https://github.com/samchon/typescript-stl> |
| **출품작 소개 (요약)** | 삼촌 프레임워크는 OON (Object Oriented Network) 프레임워크로써, 네트워크 시스템을 객체지향적으로 구성할 수 있게 해 주는 프레임워크입니다. 자주 쓰이는 네트워크 시스템을 모듈화시켜 코드 재사용성을 극대화할 수도 있습니다.  TypeScript 와 C++ 에서 사용할 수 있습니다. |

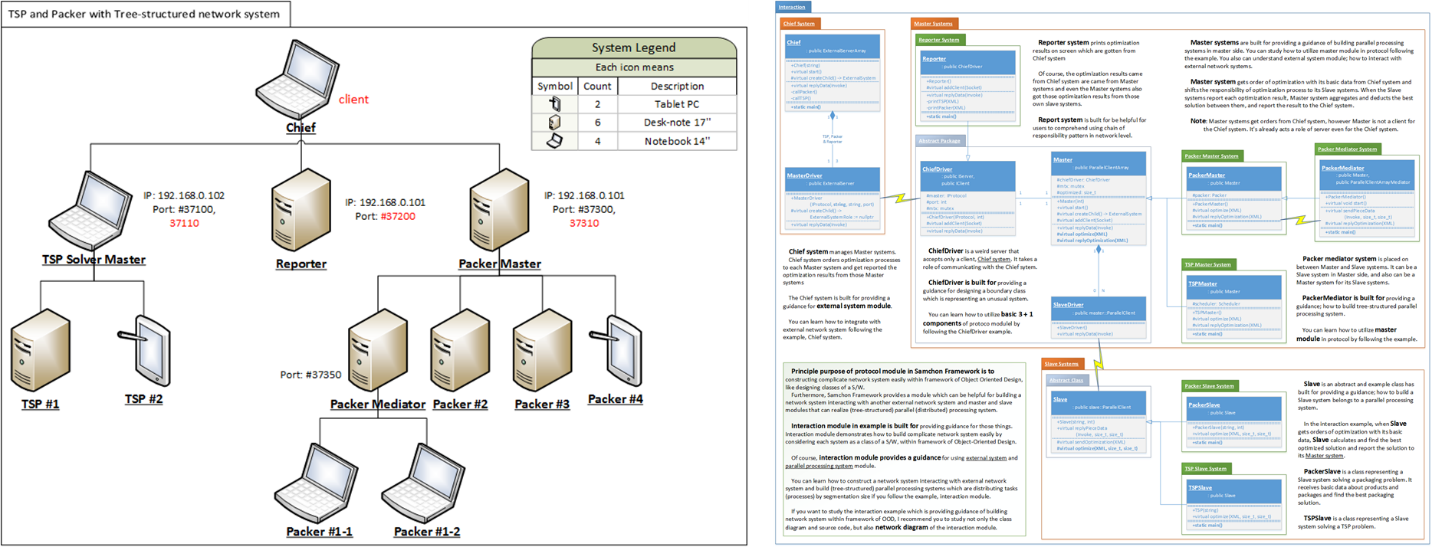
# 개발배경 및 목적

Samchon Framework 는 네트워크 시스템을 S/W, 객체지향적 (Object-Oriented) 인 관점에서 구현할 수 있게 해 주는, 객체지향네트워크 (OON, Object Oriented Network) 프레임워크입니다.

주식 시뮬레이션과 시스템 트레이딩에 관한 클라우드 솔루션 (Samchon Simulation) 을 4년간 만들어 왔습니다. 클라우드 서버니 분산-병렬처리시스템이니 하는 것들을 설계하고 구현하면서, 모종의 필요성을 느껴 제작하게 된 것이 바로 이 Samchon Framework 입니다. 나름 완성도와 체계성에 자신이 있어, 이렇게 오픈소스로 공개하게 되었습니다.

다음은 삼촌 프레임워크의 활용 방안 및 목적입니다.

## 네트워크 시스템을 S/W, 객체지향적인 관점에서 구현



Samchon Framework 는 네트워크 시스템을 객체지향적인 관점에서, 마치 S/W 오브젝트를 다루듯이, 구축할 수 있게 해 줍니다. 여타 시스템과의 네트워크 통신이나 연동같은 것은 생각하지 않으셔도 됩니다. 네트워크 통신과 연동은 Samchon Framework 가 알아서 해 드립니다. 내 시스템과 네트워크를 통해 연동되는 여타 시스템들, 이들은 그냥 **객체**입니다.

내가 구성하려는 네트워크 시스템이, 각 시스템 노드간에 그 얼마나 복잡한 연결관계를 지니던, 각 시스템 노드는 그냥 하나의 객체입니다. 객체에서 객체로, 그저 타 객체의 함수를 부르고 파라미터를 전달하듯, 그렇게 만들어 나가시면 됩니다.

## 시스템 템플릿, 재사용성의 극대화



반복컨데, Samchon Framework 는 “객체지향 네트워크 프레임워크” 입니다. 네트워크 시스템을 객체지향적인 관점에서 구현할 수 있으니 응당, 자주 쓰이는 네트워크 시스템들을, 모듈화 (템플릿화) 하여 언제든 재사용하는 것 또한 가능합니다. 이 것이 바로 **System Template** 입니다.

클라우드 시스템이 필요하면 Cloud Service 이라는 시스템 템플릿을, 분산-병렬처리시스템을 구축해야 한다면 Distributed System 이라는 시스템 템플릿을 가져다 쓰시면 됩니다. 여러분이 자주 쓰시는 특유의 네트워크 시스템 구조가 있다면, 그 또한 모듈화 (템플릿화) 하여 재사용 하십시오.

아래는 Samchon Framework 에서 기본적으로 제공되는 System Template 의 목록입니다.

* Cloud Service: 클라우드 서버 템플릿
* External System: 1: N (일대다) 의 네트워크 시스템 제어에 관한 템플릿
  + Parallel System: 병렬처리시스템 템플릿
  + Distributed System: 분산-병렬처리시스템 템플릿

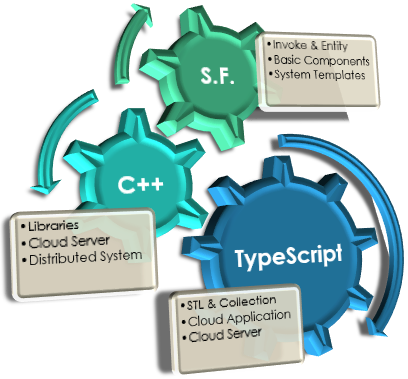
## C++ 과 TypeScript (JavaScript) 를 연동한 통합시스템 구축

Samchon Framework 는 두 가지 언어, C++ 과 TypeScript 를 지원합니다.

TypeScript 는 높은 생산성을 자랑하되, 성능에서 열위에 있습니다. 반대로 C++ 은 생산성에서 떨어지나 성능에서 우위를 보입니다. 이렇게 서로 다른 특색을 보여주는 두 언어를, Samchon Framework 는 다음과 같이 활용하기를 권장합니다.

* **빠른 런칭**
  + TypeScript 와 HTML 로 클라우드 어플리케이션 (프론트, UI) 를 제작
  + TypeScript 와 NodeJS 로 클라우드 서버를 제작
* **퍼포먼스 개선**
  + 무거운 작업을 클라우드 서버 (NodeJS) 에서 C++ 시스템에 분산하여 처리
  + TypeScript 로 만들어진 클라우드 서버의 기능을 조금씩 C++ 로 이관시켜나감

### TypeScript-STL

[](https://github.com/samchon/framework#interaction)서로 다른 언어로 만들어진 이종 시스템을 통합하고 마이그레이션함에 있어, 두 언어와 시스템이 같은 문법과 설계를 공유한다면, 그보다 좋을 수가 없습니다. 그리고 그보다 좋을 수 없을 바로 그 방안을, Samchon Framework 는 제공합니다.

같은 설계는 바로 Samchon Framework 의 객체지향네트워크 패러다임을 이용하는 것, 그리고 같은 문법은 **TypeScript-STL** 이라는 서브 프로젝트를 통해 지원합니다.

TypeScript-STL 은 C++ 의 STL (Standard Template Library) 을 TypeScript 로 마이그레이션 한 프로젝트입니다. C++ STL 의 컨테이너와 알고리즘 일체를 TypeScript 에서 사용할 수 있습니다.

이로써, Samchon Framework 를 사용하면, C++ 과 TypeScript 이 두 언어는 설계 뿐 아니라, 세부 문법과 자료구조 및 알고리즘까지 통일해 사용할 수 있습니다. TypeScript 를 이용하여 빠르게 런칭한 서비스를, C++로 마이그레이션함에 있어 어려울 것은 하나 없습니다. 그 반대 역시 마찬가지이구요.

# 개발환경 및 개발 언어

## 개발 환경

* Windows, Mac OS, CentOS:
  + 개발 및 서버 컴퓨터들의 운영체제.
  + 단, 삼촌 프레임워크는 운영체제에 상관없이 모두 이용이 가능합니다.
* Visio: 아키텍처 설계
  + Conceptual Diagram
  + C++ Class Diagram
  + TypeScript Class Diagram
  + Sequence Diagram
* Visual Studio Community: 코드 편집

## [Supported Languages Diagram](https://github.com/samchon/framework#interaction)개발 언어

### C++

* Samchon Framework 제작
* 각종 예제 제작

### TypeScript (JavaScript)

* TypeScript-STL 제작
* Samchon Framework 제작
* 각종 예제 제작

### 활용 모형

삼촌 프레임워크는, 각 언어를 다음과 같이 활용하는 것이 가장 이상적이라 봅니다.

* **TypeScript, 빠른 런칭**
  + HTML 로 클라우드 어플리케이션 (프론트, UI) 를 제작
  + NodeJS 로 클라우드 서버를 제작
* **C++, 퍼포먼스 개선**
  + 무거운 작업을 클라우드 서버 (NodeJS) 에서 C++ 시스템에 분산하여 처리
  + TypeScript 로 만들어진 클라우드 서버의 기능을 조금씩 C++ 로 이관시켜 나아감

# 모듈 구성 및 아키텍처

## TypeScript-STL



**가이드 문서**: <https://github.com/samchon/framework/wiki/TypeScript-STL>

TypeScript-STL 은 C++ 의 STL (Standard Template Library) 을 TypeScript 로 마이그레이션 한 모듈입니다. C++ STL 의 컨테이너와 알고리즘 일체를 TypeScript 에서 사용할 수 있게 해 줍니다.

TypeScript-STL 은 TypeScript 와 C++, 각기 다른 언어로 만들어진 시스템의 통합을 돕고자 만들었습니다. C++ 과 같은 문법과 자료구조, 그리고 알고리즘을 공유함으로써, 시스템 통합이나 마이그레이션을 행함에 있어, 특히 우위를 보입니다.

TypeScript 와 C++ 간에 설계를 공유하는 일은, 삼촌 프레임워크와 OON (Object Oriented Network) 디자인 패턴을 이용하시면 됩니다.

## Library



**가이드 문서**

* <https://github.com/samchon/framework/wiki/TypeScript-Library>
* <https://github.com/samchon/framework/wiki/CPP-Library>

Library 는 각종 유틸리티 클래스 및 라이브러리들을 모아놓은 모듈입니다.

* 공통
  + XML
  + StringUtil
* TypeScript
  + EventDispatcher
  + FileReference
* C++
  + HTTP Service
  + SQL Driver: SQL, DBMS 인터페이스
  + Genetic Algorithm: 유전자 알고리즘에 관한 템플릿 라이브러리
  + Critical Section

## Protocol

### Basic Components



**가이드 문서**

* <https://github.com/samchon/framework/wiki/TypeScript-Protocol-Basic_Components>
* <https://github.com/samchon/framework/wiki/CPP-Protocol-Basic_Components>

삼촌 프레임워크는 네트워크 시스템을 객체지향적으로 구성할 수 있게 해 주는 객체지향 네트워크 프레임워크 (OON, Object Oriented Network) 입니다. 그리고 **Basic Components** 는, 네트워크 시스템을 객체지향적으로 구성할 수 있게 해 주는 최소 단위의 모듈입니다.

Basic Components 에 속한 각 원소 (클래스) 들의 배합으로 네트워크 시스템을 구성하십시오.

그 어떠한 네트워크 시스템도, 그것이 제 아무리 복잡하고 거대한 구조를 가지고 있다 하더라도, Basic Components 의 조합과 활용으로 구성할 수 있습니다. 삼촌 프레임워크에서 제공하는 각종 System Templates 또한, 이 Basic Components 의 배합으로 구성된 하나의 집합일 뿐입니다.

### Standard Messages



**가이드 문서**

* <https://github.com/samchon/framework/wiki/TypeScript-Protocol-Standard_Message>
* <https://github.com/samchon/framework/wiki/CPP-Protocol-Standard_Message>

Standard Message 모듈은, 삼촌 프레임워크의 데이터 표기법과 표준화에 관한 모듈입니다.

1. **Invoke**

Invoke 는 삼촌 프레임워크의 네트워크 통신에서 쓰이는 표준 메시지 구조 (메시지 프로토콜) 입니다.

함수의 형태를 본땐 XML의 구조를 취하고 있으며, 실제로도 함수처럼 동작합니다. 여러분이 삼촌 프레임워크를 사용하는 내내, 지긋지긋하게 보시게 될, 가장 빈도수 높게 출현하는 객체, 메시지 프로토콜입니다.

1. **Entity**

Entity 는 데이터 클래스와 XML 간 상호 변환을 도와주는 클래스입니다.

EntityGroup 이라는 템플릿 클래스를 사용하여, composite relationship 을 표현할 수 있습니다.

## System Templates

### Cloud Service



**가이드 문서**

* <https://github.com/samchon/framework/wiki/TypeScript-Protocol-Cloud_Service>
* <https://github.com/samchon/framework/wiki/CPP-Protocol-Cloud_Service>

클라우드 서버를 구성할 때 사용할 수 있는 템플릿입니다.

지극히 직관적으로 클래스들을 구성해 놨습니다. 이들을 상속해 클라우드 서버를 구축하십시오.

* Server: 문자 그대로의 서버, 클라우드 서버를 의미
* User: 같은 세션 아이디를 가지는 Client 객체들의 집합
* Client: 클라이언트와 메시지 통신을 담당하는 객체. Service 와의 역할 혼용에 주의할 것
* Service:
  + 클라이언트가 요청한 path 에 따라 다르게 생성됨
  + 하나의 웹 페이지, 개별 서비스를 대표하는 객체

### External System



**가이드 문서**

* <https://github.com/samchon/framework/wiki/TypeScript-Templates-External_System>
* <https://github.com/samchon/framework/wiki/CPP-Templates-External_System>

1: N, 다수의 네트워크 시스템과 연동을 해야할 때 사용할 수 있는 템플릿입니다.

외부 시스템과 연동을 담당하는 External System 객체는 어떠한 역할을 수행하는 가를 의미하는 ExternalSystemRole 객체를 지닙니다. 그리고 이 ExternalSystemRole 은 External System 을 통해 네트워크 통신을 대리할 수 있으며, ExternalSystemArray 를 통해 직접 access 가 가능합니다. 이를 통해, 어떠한 시스템이 무슨 역할을 하는 지에 개의치 않고, 순수하게 현재 필요한 역할에 바로 접근하여 외부 시스템과의 연동을 행할 수 있는, Proxy Pattern 을 사용할 수 있습니다.

이후, 병렬처리시스템을 구축할 때 쓰이는 Parallel System 템플릿과 분산처리시스템을 구축할 때 쓰이는 Distributed System 템플릿 모두가 이 External System 템플릿을 확장하여 구현하였습니다.

### Parallel System



**가이드 문서**

* <https://github.com/samchon/framework/wiki/TypeScript-Templates-Parallel_System>
* <https://github.com/samchon/framework/wiki/CPP-Templates-Parallel_System>

병렬처리시스템을 구성할 때 사용할 수 있는 템플릿입니다. 단순 1: N 병렬처리시스템 뿐 아니라, 트리 구조의 병렬처리시스템 또한 구성할 수 있습니다. 더불어, 본 Parallel System 모듈은 External System 모듈을 상속하여 만들어진 바, 병렬처리 외에 External System 모듈 고유의 기능과 Proxy Pattern 모두 사용 가능합니다.

병렬 프로세스의 분할은, 각 슬레이브 시스템 (노드) 의 성능 지수 (performance index)[[1]](#footnote-1) 에 근거하여 이루어집니다. 이 성능 지수는 매 병렬 프로세스가 행해질 때마다, 각 노드에 할당된 프로세스의 단위 수와 수행 시간에 근거하여 상시 재평가됩니다.

### Distributed System



**가이드 문서**

* <https://github.com/samchon/framework/wiki/TypeScript-Templates-Distributed_System>
* <https://github.com/samchon/framework/wiki/CPP-Templates-Distributed_System>

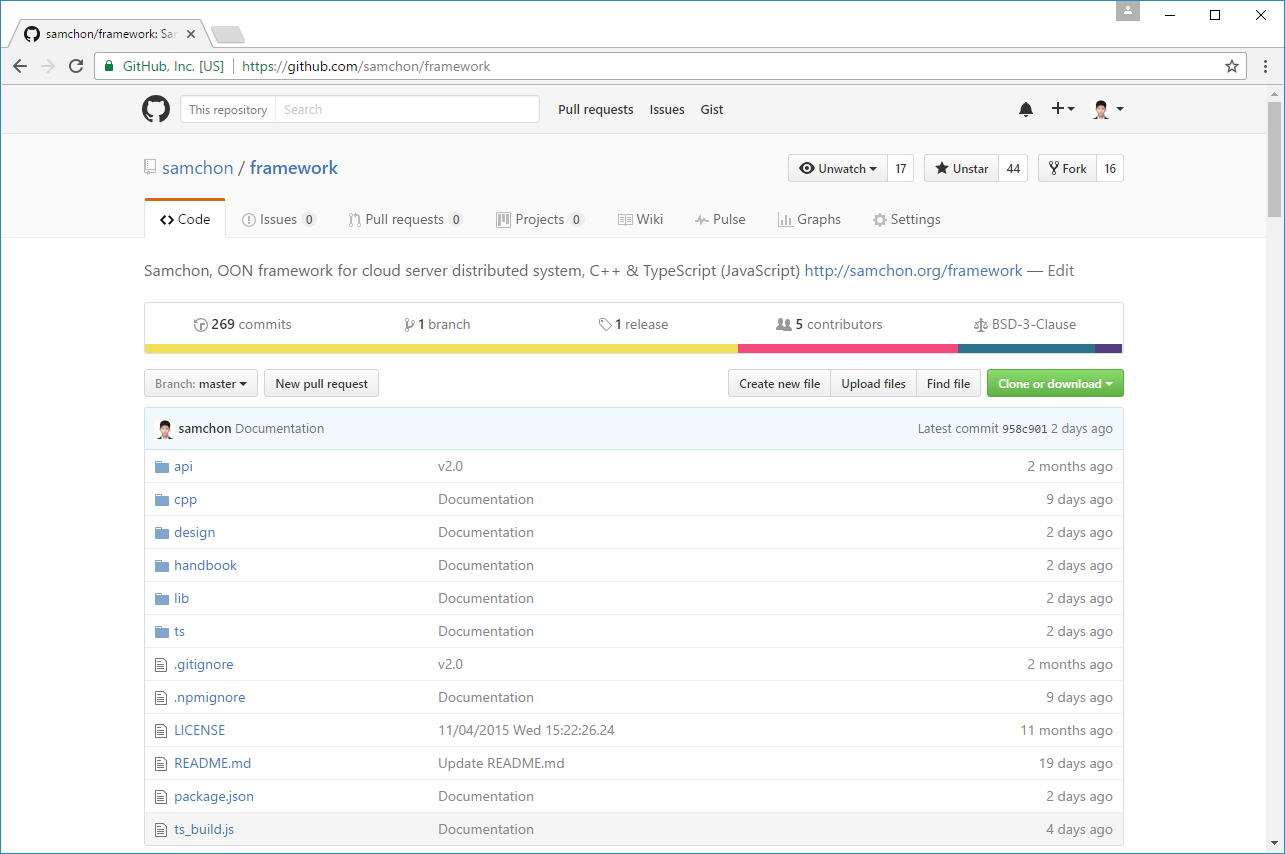
분산처리시스템을 구성할 때 사용할 수 있는 템플릿입니다. 트리 구조의 분산처리시스템 또한 구성이 가능합니다. 또한, 본 Distributed System 모듈은 Parallel System 모듈을 상속하여 만들어진 바, 분산처리 외에 병렬처리 또한 수행할 수 있습니다.

분산 프로세스의 수행은, 현존하는 슬레이브 시스템 (노드) 中 가장 유휴상태에 있는 노드를 선택하여 이루어지게 됩니다. 단, 이 때 유휴상태에 대한 판별은, 각 노드가 현재 수행 중인 프로세스의 개수 뿐 아니라, 각 노드의 성능 지수 (performance index) 와 수행코자 하는 프로세스의 리소스 지수 (resource index)[[2]](#footnote-2) 를 종합적으로 고려하여 이루어집니다. 그리고 위 지수들은 매 분산 프로세스마다 상시 재평가됩니다.

# 프로젝트 구성요소

## 코드 공개

### GitHub



* <https://github.com/samchon/framework>
* <https://github.com/samchon/typescript-stl>

Samchon Framework 는 BSD 라이선스를 따르는 오픈소스 프로젝트입니다. 누구나 자유롭게 이용할 수 있으며, 상업적인 활용에도 따르는 제약이 없습니다. 다만 재배포시, Samchon Framework 를 사용했다는 것과 원저자만은 꼭 명기해주십시오.

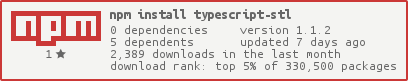
Samchon Framework 및 TypeScript-STL 의 소스 코드는 GitHub 에 공개되어 있습니다. 수 년 간의 꾸준한 업데이트와 테스트 및 배포를 거치며, 지속적으로 발전해나가고 있습니다. GitHub 에서 소스코드나 배포버전을 다운받은 후, 제공되는 설계도와 가이드 문서를 참고하시어 이용해 주십시오.

### 패키지 설치

#### C++

깃허브에서 제공되는 릴리즈 버전을 받은 후, 이용하시면 됩니다.

#### NPM Package

**[](https://nodei.co/npm/typescript-stl)**

Samchon Framework 와 TypeScript-STL 은 NPM module 에 등록되어있습니다.

간단한 명령어로 삼촌 프레임워크를 설치, 바로 이용하실 수 있습니다.

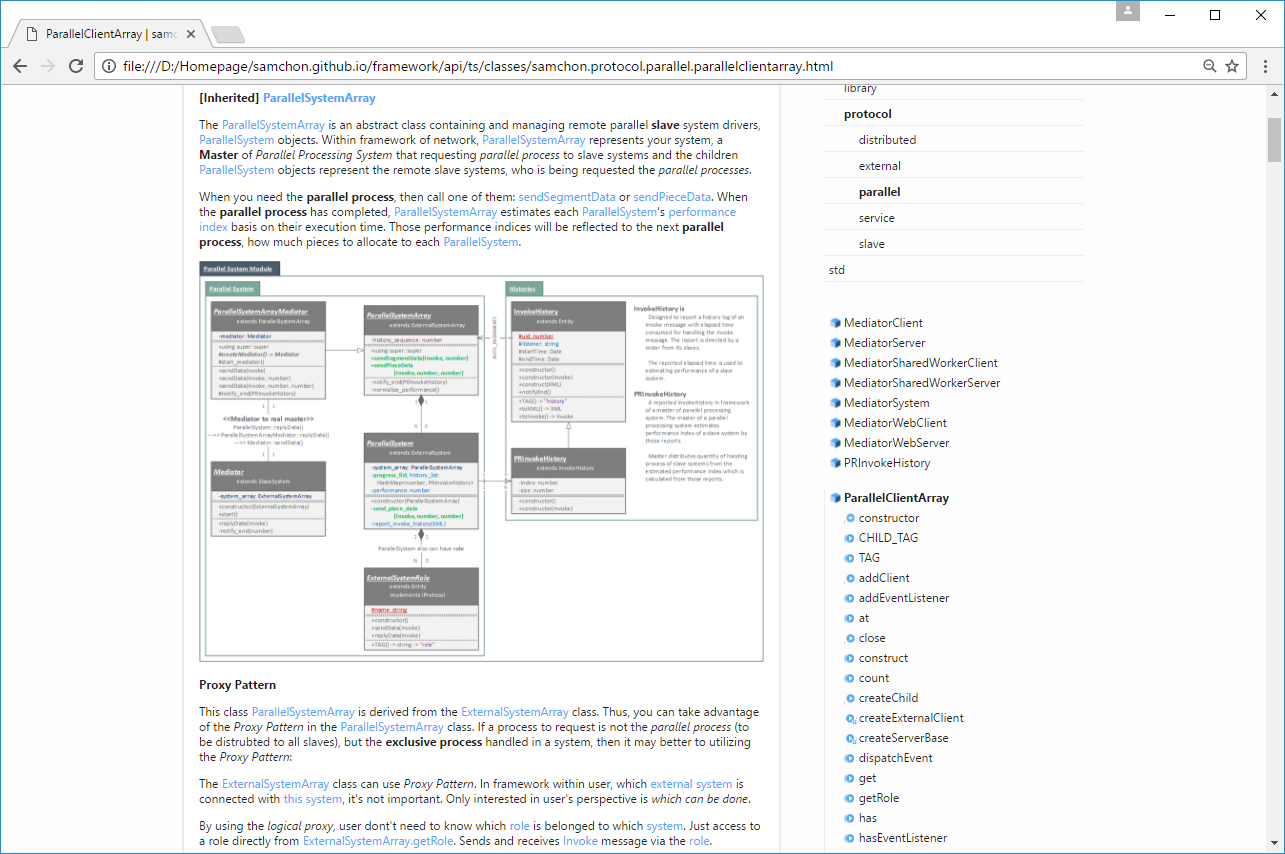
* npm install -g samchon-framework
* npm install -g typescript-stl

#### Typings & Nuget

[](http://definitelytyped.org/)TypeScript 헤더 파일 또한, 간단한 명령어로 설치하실 수 있습니다.

* NPM Types
  + npm install --save @types/samchon-framework
  + npm install --save @types/typescript-stl
* Typings
  + typings install samchon-framework
  + typings install typescript-stl
* Nuget
  + install-package samchon-framework.TypeScript.DefinitelyTyped
  + install-package typescript-stl.TypeScript.DefinitelyTyped

## 문서화



누군가 저에게 Samchon Framework 의 가장 큰 장점을 무엇이냐 묻는다면, 저는 단 한 치도 망설이지 않고 **“문서화”** 라고 대답할 것입니다. 그만큼 Samchon Framework 의 문서화는 치밀하고 체계적이며 방대합니다.

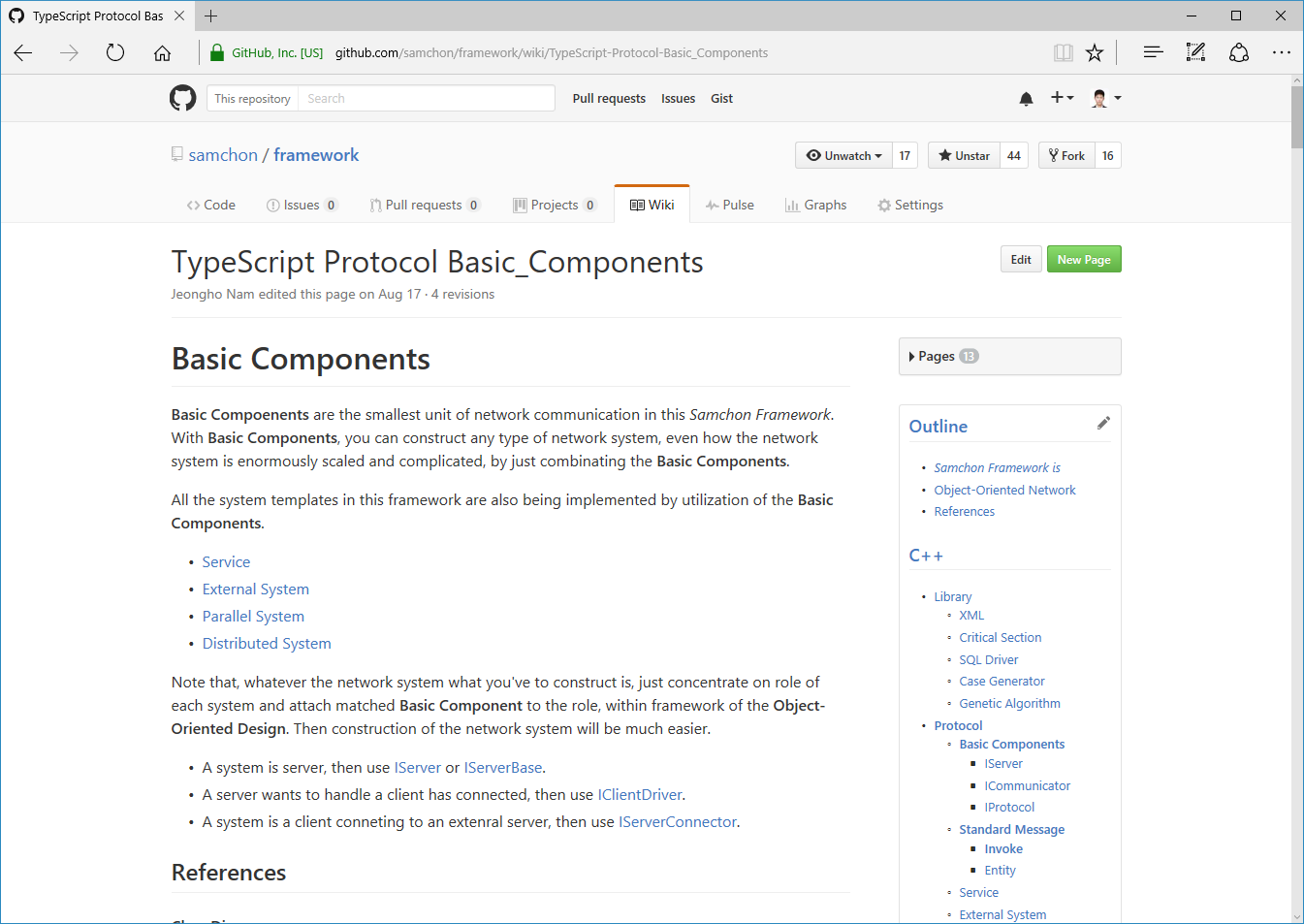
누구라도 Samchon Framework 가 제공하는 가이드 문서와 예제를 따라가다 보면, OON (Object Oriented Network) 에 대해 심도 깊은 깨달음을 얻어, 이를 자유자재로 구사할 수 있습니다. 또한, 아키텍처 설계도를 체계적이고 명확하게 정의하였기에, 각 모듈과 오브젝트간의 유기적인 관계를 쉬이 파악할 수 있습니다.

그리고 마지막으로, 약 3,000 여 페이지에 이르는 방대한 API 문서를 통하여, Samchon Framework API 의 세세한 사항까지 서술을 해 놓았습니다. 때문에, 꼼꼼한 사용자들은 Samchon Framework 에 대하여 그 무엇 하나 놓치지 않고 알차게 활용할 수 있습니다.

### 아키텍처 디자인

* TypeScript Class Diagram: <http://samchon.github.io/framework/design/ts_class_diagram.pdf>
* C++ Class Diagram: <http://samchon.github.io/framework/design/cpp_class_diagram.pdf>
* Sequence Diagram: <http://samchon.github.io/framework/design/sequence_diagram.pdf>
* Conceptual Diagram: <http://samchon.github.io/framework/design/conceptual_diagram.pdf>

### 가이드 문서

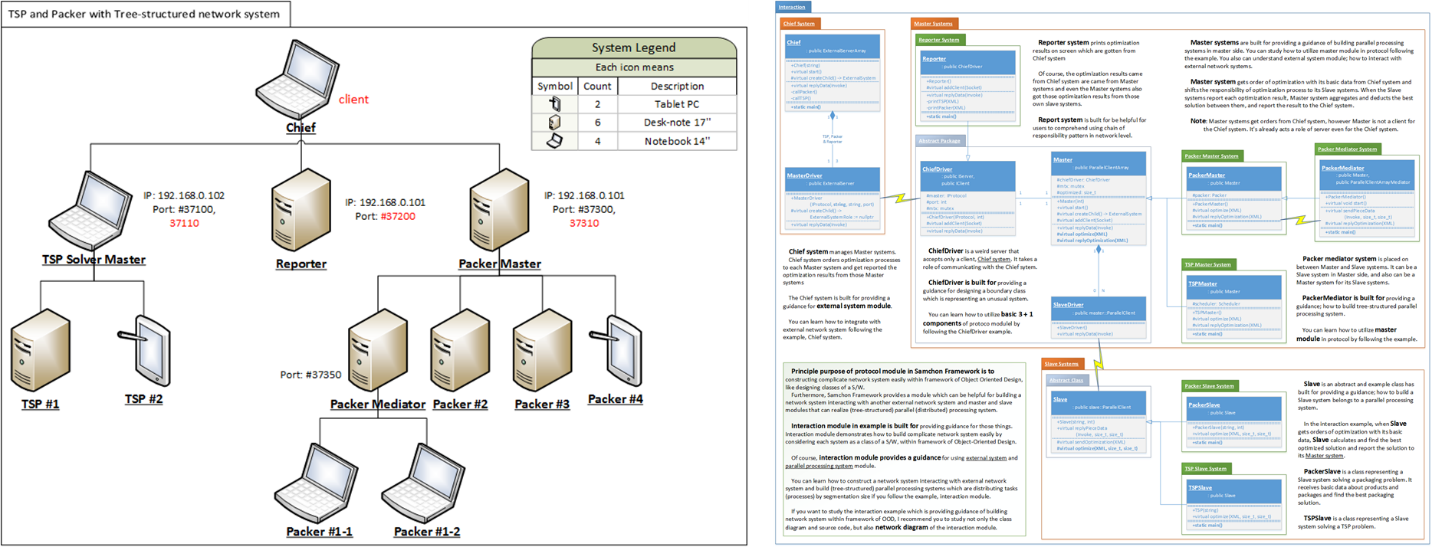


* <https://github.com/samchon/framework/wiki>

### API 문서

* TypeScript: <http://samchon.github.io/framework/api/ts>
* C++: <http://samchon.github.io/framework/api/cpp>

## 예제 프로젝트



### Chatting

* 주제: 채팅 서버 및 어플리케이션
* 사용 템플릿: Cloud Server
* 가이드 문서: <https://github.com/samchon/framework/wiki/Examples-Chatting>

### Interaction

* 주제: 트리 구조의 분산처리시스템 및 모니터링 어플리케이션
* 사용 템플릿
  + External System
  + Parallel System
  + Distributed System
  + Cloud Service
* 가이드 문서: <https://github.com/samchon/framework/wiki/Examples-Interaction>

### Samchon Simulation Cloud

* 주제: 주식 시뮬레이션 및 시스템 트레이딩에 관한 클라우드 솔루션
* 사용 언어: C++, Flex
* 주요 특징:
  + Samchon Framework 의 사실상 모든 모듈이 사용됨
  + 단, 시각화에는 TypeScript 가 아닌 Flex 가 쓰임.
  + 어쨌든 연산 작업은 C++ 이 수행
* 가이드 문서: <https://github.com/samchon/framework/wiki/Examples-Simulation>

# 기대효과 및 활용분야

## 그간의 활용

### Samchon Simulation Cloud

* 주제: 주식 시뮬레이션 및 시스템 트레이딩에 관한 클라우드 솔루션
* 사용 언어: C++, Flex
* 주요 특징
  + Samchon Framework 각 모듈의 종합적인 활용
  + 99.9 % 의 모듈이 쓰임
* 시연 영상
  + <https://youtu.be/8O5Wf0Z3sqI?t=2m3s>
  + <https://youtu.be/oKk6qwwWUM8?t=7m7s>
* 매뉴얼: <http://samchon.org/portfolio/Samchon_Simulation/Manual-KR.pdf>
* 데모: <http://samchon.org/simulation>

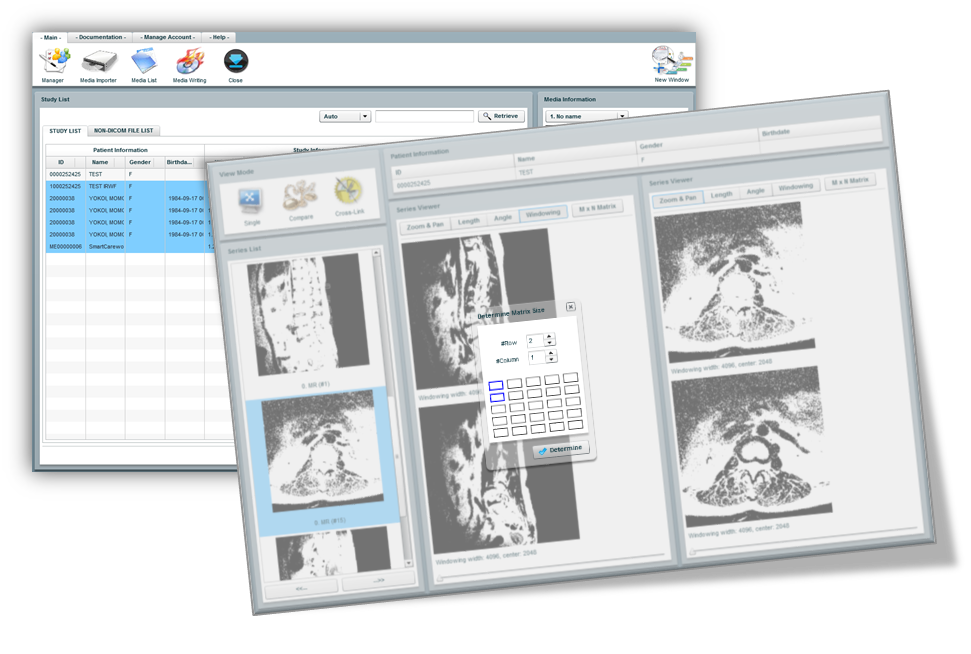
Samchon Simulation Cloud 는 주식 시뮬레이션 및 시스템 트레이딩에 관한 클라우드 솔루션입니다.

* 시세 조회
  + 주가 및 요약정보 조회
  + 재무정보 및 지표 조회
  + 상대주가 및 재무지표 비교
* 종목 검색
  + 남트리를 이용한 다차원 검색조건 생성
* 시뮬레이션
  + 백 테스팅과 남트리 및 AI 기법을 이용한 최적의 거래조건 도출
    - 이렇게 도출된 거래조건으로 시스템 트레이딩을 실시
  + 몬테카를로 시뮬레이션

Samchon Simulation Cloud 는 Samchon Framework 를 활용한 가장 모범적인 사례이자, 궁극의 예제 프로젝트입니다. 기본적으로 주식 시뮬레이션이나 시스템 트레이딩은, 그 로직이나 알고리즘 자체가 상당히 어렵습니다.

여기에 클라우드 시스템이니, 분산처리시스템이니 네트워크 연동이니 하는 것들이 덕지덕지 붙기 시작하면, 제작 난이도가 겉잡을 수 없이 높아져 버립니다. 이 때, 네트워크 시스템 구축에 대한 부담을 덜어버리기 위하여, 즉 비즈니스 로직과 네트워크 로직을 분리키 위하여 Samchon Framework 를 제작, 활용하게 됩니다.

### OraQ, Selene TR-1



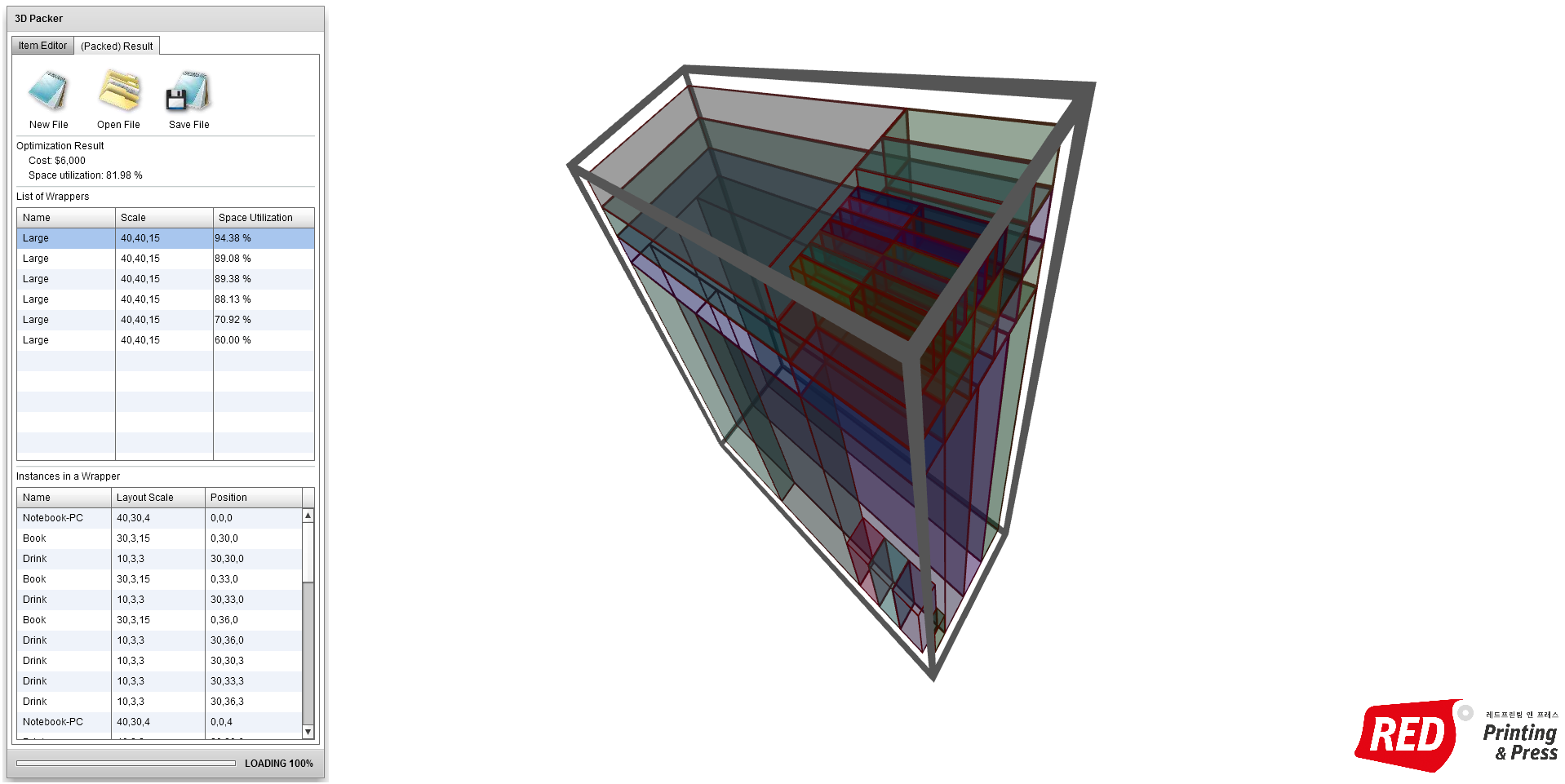
* 주제: 클라우드 PACS 지원시스템
* 사용 언어: C++, Flex, T-SQL
* 주요 특징
  + Cloud Service 및 External System 템플릿 사용
  + 표준 데이터 클래스, Entity 의 적극적인 활용

OraQ, Selene TR-1 은 클라우드 기반의 PACS 지원 시스템입니다. 제가 대학교를 졸업한 후 처음 달려든 프로젝트이며, Samchon Framework 를 활용한 최초의 상업 프로젝트이고, 가장 큰 성과를 낸 프로젝트이기도 합니다.

여타 Samchon Framework 로 만들어진 프로젝트들이 그러하듯, OraQ 또한, 무거운 작업은 C++ 에서, 가볍고 프론트에 관련된 작업은 웹에서 수행합니다. 미디어 임포팅, 영상-이미지 처리나 PACS 나 MWL 같은 외부 시스템과의 연동은 C++ 서버에서, 의료 이미지 및 영상자료를 보고, 관리하는 프론트 구현은 Flex 에서 이루어졌습니다.

PACS 시스템 자체는 알고리즘이나 로직 자체가 어렵진 않으나, 네트워크 시스템 적으로 난해한 면이 있기에, 본인 이전의 개발자는 OraQ 의 개발에 실패했었습니다. 하지만, Samchon Framework 특유의 OON 패러다임으로 난해한 네트워크 시스템을 객체지향적 관점에서 간단히 구현하고, 비즈니스 로직에 집중하여 프로젝트를 성공시킬 수 있었습니다.

### 3D Bin Packing



* 주제: 클라우드 3D Bin Packing, 포장비용 최적화
* 사용 언어: C++, TypeScript
* 주요 특징
  + TypeScript 는 시각화를, C++ 은 연산을
  + TypeScript-STL 의 전격적인 활용
  + 표준 데이터 클래스, Entity 의 극한의 활용
* 시연 영상: <https://youtu.be/mLj0dJiqFTs>

3D Bin Packing 은 삼촌 프레임워크를 활용한, 전형적인 프로젝트입니다.

무거운 연산 작업 (3d bin packing) 은 C++ 시스템에서 처리하고, TypeScript 는 이 연산 결과를 받아 화면에 시각화 해 줍니다. 단, C++ 시스템이 존재하지 않을 시에는, TypeScript 스스로 3d bin packing 연산을 수행합니다 (다만, 이 경우에는 매우 느립니다).

자료구조 및 알고리즘에 TypeScript-STL 이 전격적으로 쓰였으며, C++ 과 TypeScript 간 데이터 클래스의 공유를 위해 Entity 모듈이 쓰였습니다.

## 기대하고픈 효과이자 바라는 활용분야

### OON 시대의 도래

Samchon Framework 는 네트워크 시스템을 S/W, 객체지향적 (Object-Oriented) 인 관점에서 구현할 수 있게 해 주는, 객체지향네트워크 (OON, Object Oriented Network) 프레임워크입니다. 이 컨셉과 패러다임이 제 Samchon Framework 를 넘어서서 네트워크 시스템 구축의 대세가 되었으면 합니다.

객체지향네트워크 OON, 써 본 사람은 금새 공감하며 깨닫게 됩니다. OON, 이거 정말 좋습니다. OON 이 저와 제 주변의 일부 사람들만이 쓰는 패러다임이 아니라, 보편적 패러다임이 되기를 바랍니다.

### 다양한 System Template 의 공유

앞서 서두, “개발배경 및 목적, System Templates, 재사용성의 극대화”, 에서 적었던 글입니다.

Samchon Framework 는 “객체지향 네트워크 프레임워크” 입니다. 네트워크 시스템을 객체지향적인 관점에서 구현할 수 있으니 응당, 자주 쓰이는 네트워크 시스템들을, 모듈화 (템플릿화) 하여 언제든 재사용하는 것 또한 가능합니다. 이 것이 바로 **System Template** 입니다.

클라우드 시스템이 필요하면 Cloud Service 이라는 시스템 템플릿을, 분산-병렬처리시스템을 구축해야 한다면 Distributed System 이라는 시스템 템플릿을 가져다 쓰시면 됩니다. 여러분이 자주 쓰시는 특유의 네트워크 시스템 구조가 있다면, 그 또한 모듈화 (템플릿화) 하여 재사용 하십시오.

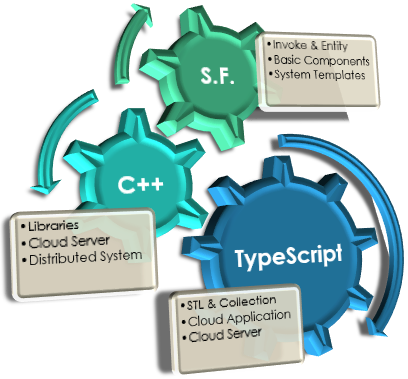
저 노란색 음영의 글귀가 현실이 되기를 바랍니다. Samchon Framework 와 함께라면, 네트워크 시스템의 각 노드를 객체화 시킬 수 있고, 네트워크 시스템 전체의 구조를 하나의 모듈로 표현이 가능합니다. 이렇게 모듈화된 System Template 이 하나의 오픈소스나 라이브러리처럼 널리 공유되고 보편화되었으면 합니다.

System Template 이 보편화되고 널리 공유되어 각종 네트워크 시스템을 보다 쉽게 구현할 수 있다면, 개발자들도 더더욱 자신들의 비즈니스 로직에 집중할 수 있게 될 것입니다. 그리고 그 속에서 Samchon Framework 도 함께 유명해졌으면 합니다.

S/W 의 유용한 알고리즘이나 자료구조가 모듈화되어 오픈소스로 널리 퍼지는 일은 너무나도 흔한 일상입니다. 이러한 일상 속에, Samchon Framework 를 이용해 만든 System Template 또한 오픈소스의 한 영역을 차지하기를 바랍니다.

### C++과 TypeScript (JavaScript) 의 통합

지금까지는 저의 목표이자 바램을 적어놓았습니다. C++ 과 TypeScript 시스템을 통합하는 데에 있어서는, Web Assembly 같은 로우 영역에서의 통합이 아니고서야, Samchon Framework 가 현존하는 프레임워크나 라이브러리 중 가장 유력한 프레임워크임을 확신합니다.

[](https://github.com/samchon/framework#interaction)서로 다른 언어로 만들어진 이종 시스템을 통합하고 마이그레이션함에 있어, 두 언어와 시스템이 같은 문법과 설계를 공유한다면, 그보다 좋을 수가 없습니다. 그리고 그보다 좋을 수 없을 바로 그 방안을, Samchon Framework 는 제공합니다.

같은 설계는 바로 Samchon Framework 의 객체지향네트워크 패러다임을 이용하는 것, 그리고 같은 문법은 **TypeScript-STL** 이라는 서브 프로젝트를 통해 지원합니다.

TypeScript-STL 은 C++ 의 STL (Standard Template Library) 을 TypeScript 로 마이그레이션 한 프로젝트입니다. C++ STL 의 컨테이너와 알고리즘 일체를 TypeScript 에서 사용할 수 있습니다.

이로써, Samchon Framework 를 사용하면, C++ 과 TypeScript 이 두 언어는 설계 뿐 아니라, 세부 문법과 자료구조 및 알고리즘까지 통일해 사용할 수 있습니다. TypeScript 를 이용하여 빠르게 런칭한 서비스를, C++로 마이그레이션함에 있어 어려울 것은 하나 없습니다. 그 반대 역시 마찬가지입니다.

### 표준 IoT 프레임워크

사물인터넷의 성패도 결국에는 여타 네트워크 시스템과 전혀 다르지 않습니다. 사물 인터넷의 성패 역시 각 사물 (노드) 간의 연동에 달려있습니다. 보편적인 프르토콜을 정의하고, 이를 통해 각 사물 간의 유연하고도 확장에 얼마나 능한 시스템을 구성하냐가 핵심입니다.

그리고 여기에도 삼촌 프레임워크의 OON 과 System Templates 가 한 몫을 해내는 날이 오기를 손꼽아 기다리고 있습니다. 삼촌 프레임워크는 각 사물을 문자 그대로 객체적인 관점에서 다룰 수 있으며, 사물인터넷의 네트워크 시스템 구성을 모듈 단위로 설계하여, 재사용성에서 우위를 확보할 수 있습니다.

1. 성능 지수 (performance index): 해당 슬레이브 시스템이 얼마나 빠른 지를 나타내는 척도 [↑](#footnote-ref-1)
2. 리소스 지수 (resource index): 해당 프로세스가 얼마나 무거운 지를 나타내는 척도 [↑](#footnote-ref-2)