eLego – Building Blocks for Software Systems

Keith Mai, eBeam
Ananth Kasillingam, eBeam

Abstract – Software systems evolve over time and gain in complexity. It is desirable to re-use solutions that have been proven to work, but this desire is often not fulfilled as redundant solutions are found in large scale systems. An infrastructure is needed to help guide developers to minimize redundant solutions and empower system designers to build systems faster. eLego is an infrastructure that encourages re-usability, extensibility, modularity and consistency across software systems to improve time to features of software.

I. Introduction

"It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change." – Charles Darwin

The solution that wins is typically the solution that meet the customer needs early and can evolve into being the optimized solution of choice.

In the academic realm (R&D), it is often best to come up with the perfect system to solve a problem. However, in the business world, bringing the perfect solution to market alone may not win the business.

With increasing complexity in use cases, the ability adapt quickly to new use cases become a critical differentiating aspect of software systems. However, responding quickly alone may not be sufficient, the quick solution provided must be reliable and provide a path achieving optimization.

There are solutions that leverage only what's already done, while other solutions try to build everything from scratch. While both approach have merits, the more sensible approach would be a balance between these approaches. The goal should be to create a solution that meets your need in the shortest time while leaving room for improvements in the long run.

II. Problem Space

Every systems built share the following attributes: it can have inputs, output and configurations. If we can understand these attributes, all systems can be seen as a single block that takes inputs, process configuration and generate output. Figure 1 shows a generalized system.

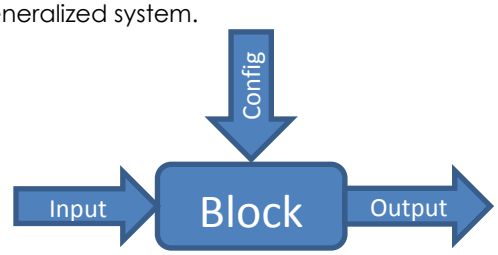

Figure 1. Generalized concept of a system

When building systems, we can speed up the design process if we leverage existing systems. One such method would be to leverage existing systems to create a new system with new inputs and outputs as shown in Figure 2.

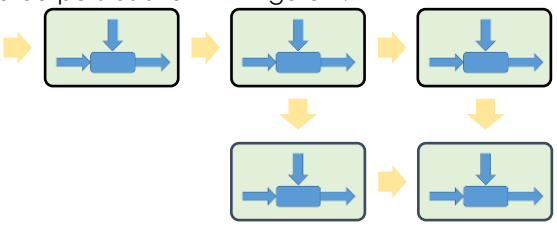

Figure 2. System of Systems

System designers may leverage this same principle and build even more complex system as shown in Figure 3.

Figure 3. Complex System of Systems

When requirements change, systems need to adapt. Systems may increase in complexity, reduce in number of operations, or sub-systems swapped out.

Figure 4. Modified Complex System

One of the biggest challenge of rapidly building new systems is the lack efficient infrastructure to leverage existing sub-systems. If such an infrastructure exists, systems can be connected together to minimize new or redundant work to produce new systems. Some common challenges are:

- Replacing subsystems
- Add-in additional subsystems
- Altering current system flows

The challenges come not from the above steps but from performing the above while maintaining system reliability, performance, and robustness.

System evolves and become more complex, this makes it challenging to manage the interaction between the system components. Since the management of interactions is already complicated, adding more components into that system can become time consuming and expensive operation. An infrastructure is needed to provide guidance on creation of subsystems and a consistent way of integrating systems together with minimal efforts.

III. Idea/Proposal

When we were young, we enjoyed playing Lego where we built complex structures with smaller pieces. eLego is an infrastructure that tries to accomplish the following principles:

- An eLego system is defined as a set of operations that takes inputs, works with configuration and generates outputs.
- Any eLego system shall be compatible with any other eLego system
- Optimization of individual eLego systems is left to the system designer, eLego's primary focus is to provide flexibility in system design.
- Connecting eLego systems shall not require code changes.
- The infrastructure shall strive to be language agnostic to enable developers to use their tool of choice.

If we consider each system defined as a single Lego block, then these systems all have pre-defined connection that can be used to connect to other blocks. Leveraging on what we did in the past, we can build larger complex systems with smaller pieces as long as we can connect the pieces together.

By employing the workflow principles, we can build an infrastructure where we will reduce development time of new systems, expedite feasibility studies and produce less erroneous systems.

IV. Data Collection Application

Today, KT tools generate an enormous amount of data in various formats and are processed in different ways. Some ways we know today, some ways we are moving away from, and some ways we are not sure what we want to do. In order to have our data collection become less coupled to data processing, we need to

break up the data generation and data processing.

Figure 5. Typical data generation and process

The goal here is to decouple data generation and data processing. All data generated can be submitted to a common medium while the processing of data will be defined elsewhere. To support existing data already generated, a common and simple interface was created to allow easy interaction with the eLego infrastructure. This interface is defined to be a Data Warehouse which takes any data object and can route the data to any processing blocks (existing or future blocks) that is unknown to the data generator.

Figure 6. Existing data generation with eLego

The data warehouse processing of the data is completely decoupled from the data generator; however, there is one dependency, the data generator needs to provide the data warehouse with the logic to convert its data object into a common object defined by the data warehouse.

A generic Translate eLego block is provided to help translate any data object to a common structure which will be used for processing. The translation block is defined so that new logic to work with any new data objects can be added at runtime, this enables the translate block to be extensible for future objects without jeopardizing current data object handling.

As all data objects will be translated to a common object model, data processing can be implemented without concerns of data generation. Data processing blocks are created to process data and generate results that suit the required needs. In this case study, calibration data were written to files with customized format defined by the processor. This logic is solely contained inside the processing logic. As new processing logics are created, they are added to the processing block. New data generated will be able to leverage any of the existing data processor.

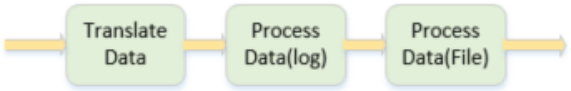

Figure 7. Data warehouse default configured flow

The flow of operations are defined using the eLego infrastructure which allows interaction with any eLego blocks without changing code.

Any data can now be submitted to a data warehouse and the user does not need to define how the data will be processed at generation time. Data translations are defined on-demand.

Calibration routines were modified to submit the calibration results to the data warehouse. These calibration routine also provides translation logic to convert its data object into a common format specified by the data warehouse. The data warehouse now have the capabilities of processing the newly submitted data.

The submitted data can now be used for analysis for any form of processing available. Some samples are possible applications are:

- Trending analysis
- Tool health check
- Feedback loop
- Debugging
- FDC/KlearPoint integration

This opens up the ability to perform additional data analysis on any data generated without the cost of modifying existing code.

V. Future Considerations

As system engineers learn about eLego and its available building blocks, systems can be build and tested with existing functionalities without involvement of software engineers. System engineers can also create new blocks and have these blocks integrated into existing systems by themselves for characterizations. An example: when building a new calibration or altering existing calibration flow, system engineers can perform these tasks directly by refining the flows of those calibrations. Once the flow is refined

and characterized, the final flow can be provided to software engineer to optimize. This can save a tremendous amount of time that we spent today going back and forth between software and systems.

VI. Conclusion

By employing a common infrastructure, we are able break down systems into smaller blocks and have these blocks worked on in parallel. As the blocks are smaller, it is easier to test and produce high quality results. With the infrastructure well defined, it enables the re-usability of pre-existing blocks easily which saves a lot of time. This infrastructure will empower system engineering with the capabilities of the current building blocks to build more complex system rapidly. This helps build a modular, consistent, reliable, flexible and lean system with low coupling and high cohesion.