Foreword

This document is an extension of more generic need/want–to-have keywords/key frases into a more concrete architectural context. Some of the keywords/key frases are:

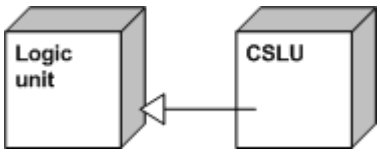
- divide and conquer
- minimal dependancies
- reusability

Terms and definitions

Logic unit:
A logic unit is a generic building block representing certain logic. The logic it represents should be independent of the context in which it is used. Each logic unit has a single core function.
An example of a logic unit is an abstract factory (see 'Design Patterns' by Gamma).

CSLU:
A CSLU (Context Sensitive Logic Unit) within this scope is a specialization of a logic unit that derives at least part of it's reason to exist from the context in which it is placed. Aside form the context knowledge all rules that apply to a pure logic unit also apply to a CSLU.
An example of a CSLU is a mesh factory. A mesh has a certain meaning within the context of rendering. The fact that the factory knows it is creating a mesh turns the factory into a CSLU.

Relationship between logic units and CSLU's:
The seperation between logic units and it's specialization CSLU is made to help purify and define toolkits of truly generic building blocks without contamination from context sensitive logic units.



Application layering

The architecture of an application can be defined as a number of layers each with it's own meaning and boundries. Such a division is not always nessesary but that doesn't mean it is not a good idea.

CSLU's are defined at different abstraction levels. Application layering is introduced to explicitly group CSLU's belonging to the same or similar level of abstraction.

Defining layers can help limit contamination within the hierachy that results from a decomposition at the lowest logical level using logic units and CSLU's.

Experience shows that contamination between CSLU's can occur over time despite the best intentions of the people involved. Application layering (among other things) allows you to introduce boundaries that help prevent the contamination from spreading to far. They act as a sort of firewall and quality portal.

Application sub-system

An application sub-system can be defined as a product capable of beeing integrated into a larger system to create an overall application system as seen by the end-user

Note that you can define multple levels within the context of application sub-systems as well. For example a platform like sub-system can serve as an application level building block for other sub-systems.

This is typically the layer that is most suited for outsourcing to a 3rd party supplier since a sub-system will typically offer a lot of functionality within it's context with relativly minor connections to the outside world. As such they are perfect candidates to be turned into plugins, allowing dynamic extention of your applications.

An example of a low level sub-system is for example a block called 'connectivity' in which an entire array of clients are defined for a wide variaty of communication protocols. This might include a URL based system with support for a HTTP client, a FTP client a POP3 client etc.

An example of a high-level sub-system would be tool plugin that is seamlessly integrated into the application adding a set of functions to the application as a whole.

Functional context

A functional context is basicly a *conceptual* sub-set of an application sub-system. It may only exist at the conceptual architectural level without a direct counterpart in a concrete design. Instead the functional context may be directly translated into a CSLU if you can avoid contaminating the CSLU with application specific context.

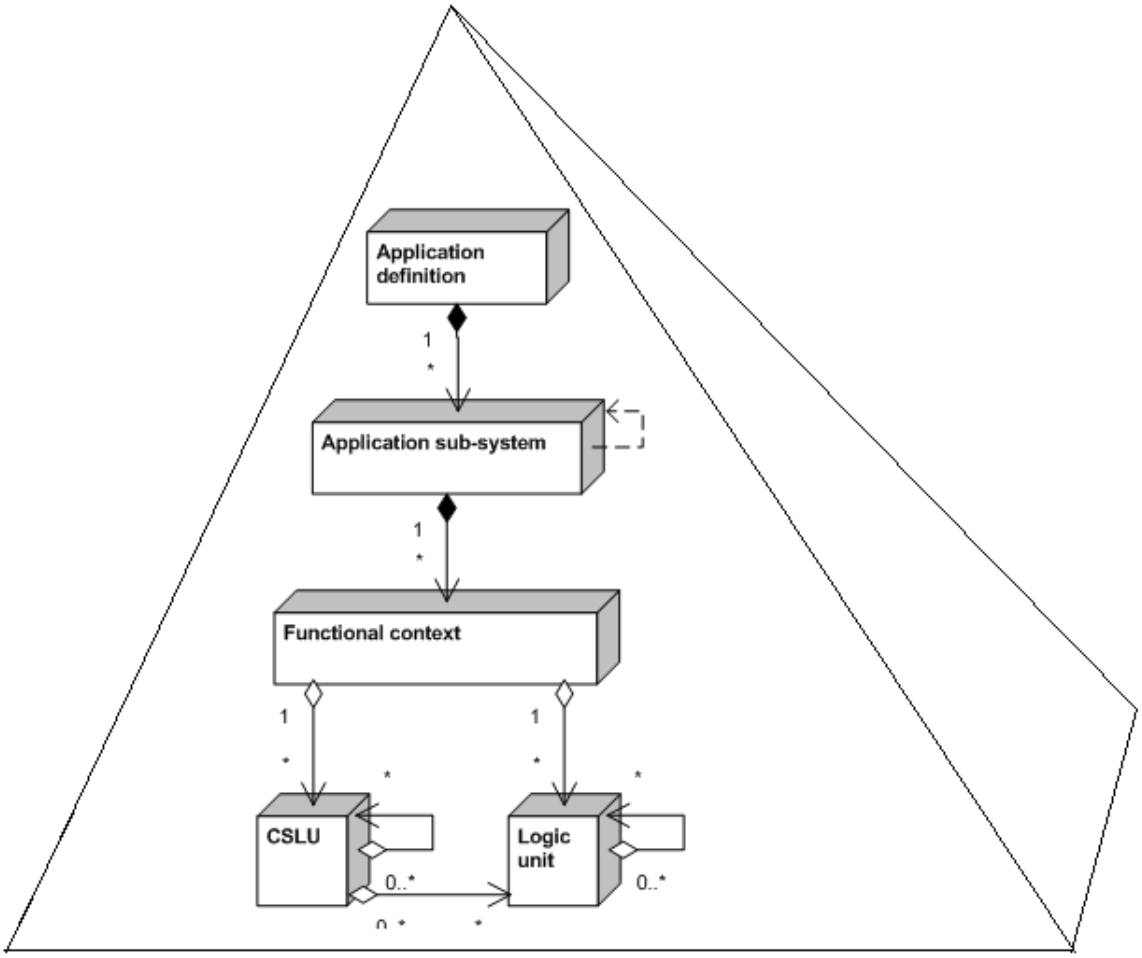
In a high-level application sub-system a functional conext is typically related to a set of use-cases at the design level. In low-level sub-systems a functional context is typically related to a set of CSLU's/Logic units that can be seen a family.

A more concrete example could be the addition of BitTorrent support in the 'connectivity' sub-system mentioned in the 'Application sub-system' segment. You might need the following CSLU's: BTClient, BTServer, BTPeer and a BTArbiter. This family of CSLU's combined can be referred to as a functional context which adds the BitTorrent functional context to the 'connectivity' sub-system.

Architectural requirements / rules

The following are a number of example requirements / rules and should by no means be considdered complete or without errors at this point. It does however give you an example of how you can refine your architectural vision further.

1. Ability to add functionality to a functional context without disrupting said context
2. Ability to add functionality to an application as a separate functional context without disrupting the application.
3. Applications are defined as a set of sub-systems where the top-most application layer acts purely as a glue layer between the relevant sub-systems.
4. Sub-system awareness is limited to the highest level application logic, there is no direct awareness of specific sub-systems between sub-systems.
5. An application sub-system is defined as a common set of application level functions belonging to the same application level context.
6. Application level functionality should be designed in terms of logic units and CSLU's. No sequential function logic is allowed, meaning that the logic units cannot be constructed in a way that forces it's user into a specific usage pattern. This makes the logic units basicly part of a sequential chain of function logic, hence the term. If function specific logic is required then it should be limited to a glue layer between the different logical components. This limitations is a requirement for the reuse of logic units as building blocks.
7. Logic units should either consist out of an atomic logical context (basicly a logic unit that is self-contained logic wise and cannot be divided further) or as the top of a pyramid/frustum comprised of logic units.
8. A logic unit that is part of a logic pyramid/frustum can only have dependancies on the same or lower level or the pyramid/frustum. Dependancies to a higher level are never allowed, such a dependancy automaticly places the logic unit at this higher level.
9. Pure logic units cannot use CSLU's, doing so turns the logic unit into a CSLU itself.
10. No matter what scope you choose for your view on the architecture it should always resemble a pyramid. Any deviation from this shape should be considdered as an error by default and should be reviewed. Note that it is 'ok' for the pyramid to resemble a frustum, this is an accepted deviation.


Application definition
1
*
Application sub-system
1
*
Functional context
1
*
1
*
*
*
CSLU
Logic unit
0..*
0..*
0..*
*