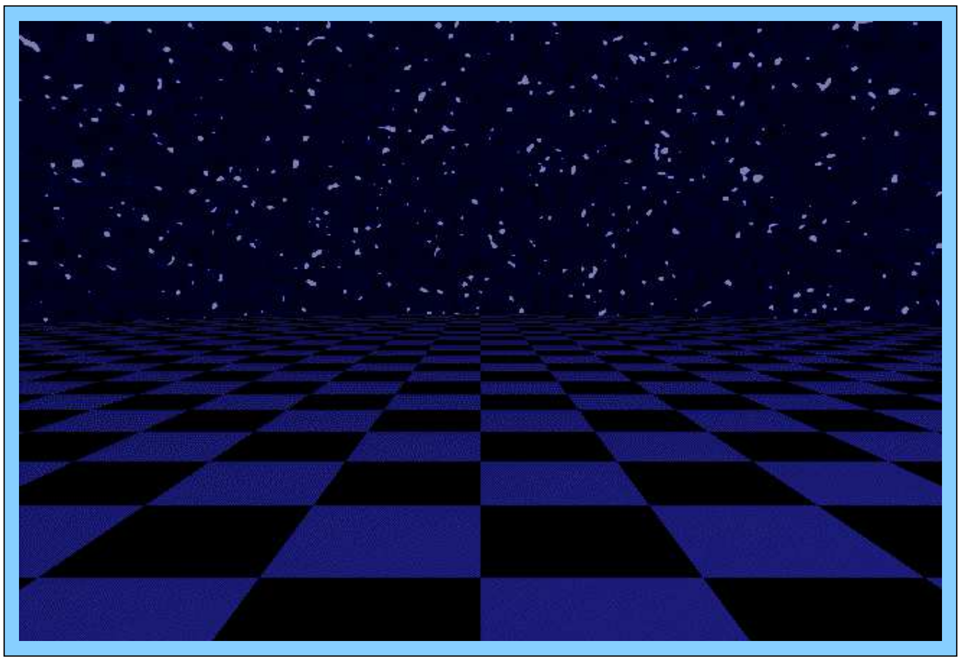
CAP

The CyberSpace Architecture Project

THE SPACE

STRUCTURING YOUR SOFTWARE SYSTEMS WITH SPACE CLASSES

*This paper describes the space related classes of the CyberSpace Foundation (*CSF*) and it's advantages in structuring, locating and moving software systems. These classes define the base of the CyberSpace Architecture (*CSA*). Specifically the classes „Area", „Frame", „Context" and „Gate" are introduced and explained. Also the relation to the CyberSpace Object Architecture (*COA*) is described, another concept of* CAP.



The CyberSpace. Actually this is only a graphical presentation of an empty (and quite boring) part of it.

1 Introduction

Every piece of software is inherently located somewhere. This is not immediately obvious, as it is extremely easy to copy, change or move such software pieces and therefore the location property is usually only perceived if problems arise.

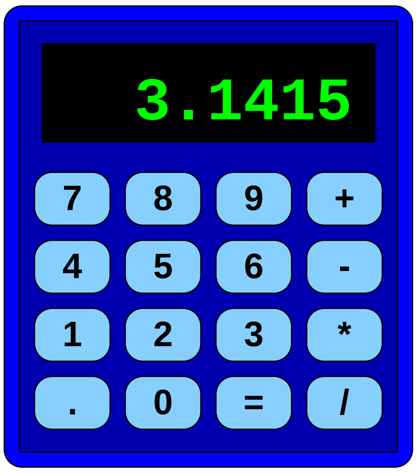
But the fact, that software is associated with it's environment has been observed since the concept of algorithms is known and is reflected in such phenomena like „environments“ or „porting problems“.

Being associated with an environment doesn't mean anything else but being located somewhere in a space, merely that this space isn't as imagenable as the physical space with it's three dimensions.

Within the CSA this space is called „CyberSpace“, a term taken originally from science fiction literature (William Gibson), which seems most appropriate to denote this space.

It should be emphasized here, that in contrast to the many different possibilities of visualizing this space, it is considered here as a purely logical concept with two main properties: it has a capacity to store information and this information has a location inside the space.

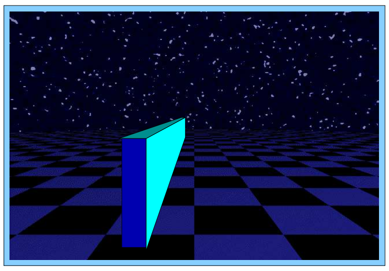
Another clarification should be made. The term „CyberSpace“ is often used for worldwide computer networks. But not the network as such has the property of a space: you cannot put anything into it. The network only connects all the computers, and they provide the space. In the context of worldwide connected computers with lots of space the perception of „space“ is much more evident than when looking at a single pocket calculator.



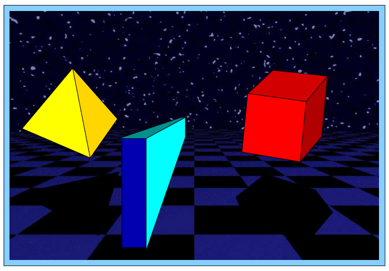
Inside you find a small piece of CyberSpace.

2 Modelling the CyberSpace

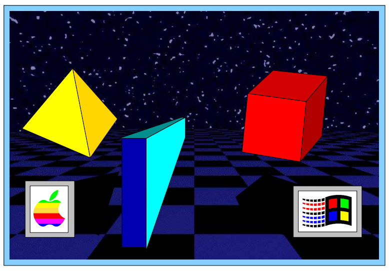
Keeping in mind this association to locations and regarding object oriented modelling techniques of software, this leads to the question of how to take benefit from this knowledge by modelling the CyberSpace. Some observations may help here.



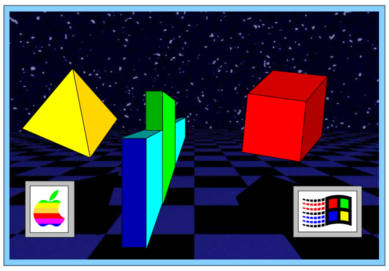
The first observation is that the CyberSpace is devided into subspaces. This division is neither complete, nor static, nor are these subspaces disjunctive. But there are a lot. The subspaces define boundaries between and around them.



Next, as mentioned above, things in CyberSpace (objects, information) have a specific location at a given time. Of course, some of these objects may be able to travel around.



Furthermore these things are not only located somewhere, but have quite different meanings depending on their location. In other words, the locations provide a specific context.

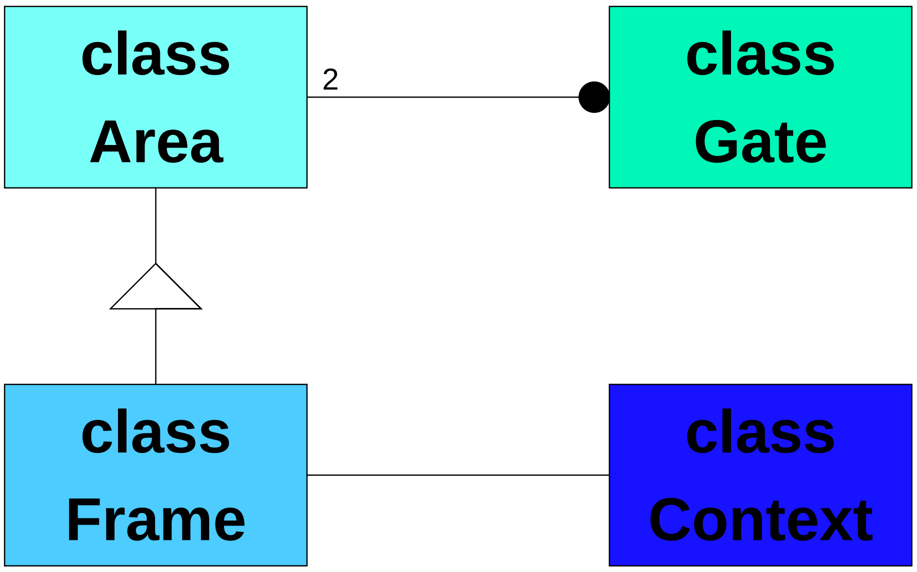


And last not least, if objects travel around in CyberSpace, they have to pass the boundaries between the subspaces. The best way to do this is to use the door.

Now, modelling these items has a lot of advantages. The model provides a means to handle them, to have a well defined interface available to access information related to them, to define a place where to put this related information and to state some things explicitely which otherwise had to be stated implicitely or, even worse, couldn't be stated.

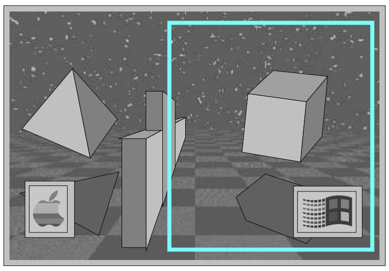
The model of the CyberSpace and it's components, which is presented in the following sections, is based on the object oriented paradigm. It is based on the above mentioned observations and consists of the classes `Area`, `Context`, `Frame` and `Gate`.

The following figure shows the class relations of the space related classes of the CSF in OMT notation. Following the classes are explained in detail.



The space related classes of the CSF *and their relations.*

2.1 Class Area



class Area

The class Area defines, as it says, areas. It doesn't make much sense to model the whole CyberSpace, but modelling areas is perfectly sensible. Areas define arbitrary subspaces of the CyberSpace. Areas can overlap, include each other, be ordered hierarchically, exist temporarily and can be very small or gigantic in size.

2.1.1 Examples of Areas

Examples of Areas include

- your computer
- a network
- all the accounts you have
- the scope of a subroutine
- all computers of a company
- all computers of a country
- all media of a country (computers, disks, etc.)
- a part of the Internet
- the subdirectory on your PC with the games

2.1.2 Kinds of Definition

All these examples have in common that there is a physical medium associated with them, i.e. a place, where you can store bits. The respective areas are defined through the limits of this medium.

Alternatively you can define an Area by specifying a constraint which must be fulfilled by any part of this Area. Each Area you can define by the location of it's medium you can also define by giving a constraint, but not vice versa.

E.g. all computers in a country are defined by the constraint that the respective Areas have to be in that country.

2.1.3 Virtual `Areas`

But still, independant of the kind of their definition, `Areas` so far mentioned are in fact realized somewhere, even if this place is no longer localized but instead spread around.

Probably this restriction is not necessary and it can be meaningful to include virtual `Areas`, like „english spoken“, „C++-language understood“ or „all working programs (including non-written ones)“.

Related to this problem are several questions, like modelling these properties as specific kinds of contexts, or whether an object in a view has identity or only represents the identity of something else and serves as a handle.

Currently the decision whether to include virtual `Areas` is not made.

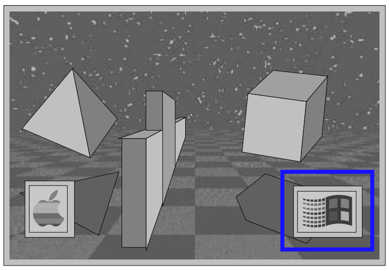
2.1.4 Proportions of `Areas`

The size of the mentioned `Areas` may vary, as you have seen from the examples given above. Also the relative sizes between `Areas`, `Frames` and `Contexts` may vary. Two examples for this shall be given.

First, consider a big `Area`, with a few interfaces and much space for objects to act inside. This may be, e.g., a big company and the interfaces are the views of this company to the world.

Second, consider a small `Area` with the inside only containing a small algorithm which interacts on the views of it's `Frame` and being very integrated with this views. This is something like a view of the world of an agent.

2.2 Class `Context`

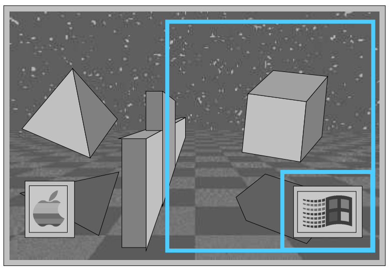


class Context

To give specific `Areas` a meaning according to their informal intended semantic, their must be a way to associate information with them and to provide a standardized way to access this information.

This information is denoted here as a `Context`. The word environment, which is often used in a similar way, is defined here as the concrete things around an area, while context is more abstract. The relation between them is a bit of a class-object relation. You can say, that an environment defines or provides a `Context`.

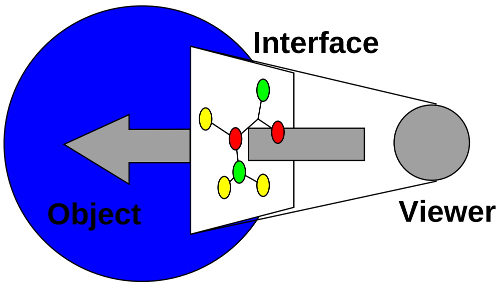
2.3 Class `Frame`



class Frame

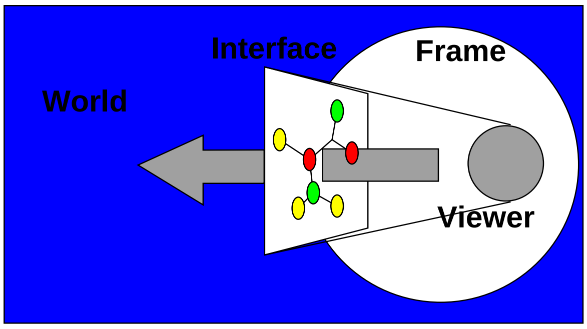
To complete the picture, a `Frame` is a subclass derived from an `Area` which provides a `Context`. An `Area` may be viewed from inside or from outside, while a `Frame` concentrates on the inside view.

A `Frame` is in a sense the contrary of an object: an object encapsulates it's inner components, providing access to them through a well defined interface.



An object encapsulates it's inner components.

In contrast to this, a `Frame` encapsulates the outside world. Access is also provided through a well defined interface.

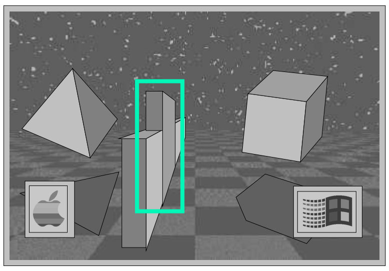


A `Frame` *encapsulates the outside world.*

In the special case of a `Frame` these interfaces may be called „Views“.

Thus, a `Frame` provides a context for algorithms or systems to allow them to be founded on a well defined base.

2.4 Class Gate



class Gate

As mentioned above, objects traveling around in CyberSpace, passing boundaries between subspaces, should use the door. This is to avoid that they have to break through the wall and get damaged or otherwise confused.

The class Gate models these orderly pathes between Areas. Today's gates can be found in form of import/export functions, data converters and e.g. automatic data transformations within file systems (compression/decompression, conversion of CR/LF sequences).

Gates can provide a lot of services. These include:

- transformation of data formats (translation between different operating systems or applications)
- translation of texts between different language Areas
- controlling access for security purposes
- translation of references (local file pointers to world wide web file pointers, inner country addresses of people to foreign country addresses)
- adapting data to specific Area constraints (moving data into a persistent Area as an abstract database storage operation)

Passing a Gate is therefore an easy to comprehend abstraction for many different operations.

3 Travelling Around in CyberSpace

Also these classes can be used with profit for common software components and operations like e.g. copying and converting files, it's power is potentiated when used in conjunction with specially prepared objects.

Within the CSA project such objects are provided by the means of a special architecture, the *CyberSpace Object Architecture* COA. COA objects can be equipped with many useful information, allowing them to be viewed under different aspects and to change their appearance.

This is espspecially useful for visualization of `Areas` and their contents under many and often unforeseen aspects. To achieve this, COA objects provide their information to a high degree in a generic fashion.

Equipped with these features, COA objects are able to interact with `Gates` and can, like a chameleon, easily adopt to the respective `Areas` they travel through and provide the services and information they are requested to provide.

The COA is an open concept. It is not necessary to convert everything to COA objects. Instead the architecture allows to embed existing things in a hull which conforms on it's outside to the COA requirements.

4 Taking Benefit

There are lots of applications of these classes and hence as many possibilities to take benefit from them:

- Modelling *physical units* like computers, devices and networks to provide technical information. This allows automatic conversion of data and adaptation to the respective local environments. It includes conversion of file formats, updating of external file references, linking to local files and parametrizations. The information provided supports porting of software and hence is another step to „plug & play" systems.

- Modelling *organizational units* like companies, departments and offices to provide contextual information. This includes again data conversion, but under different aspects like translation of textual elements to different languages or adaptations to local time and currency. Security control can be associated with the passing of `Gates`; data check in and check out in client/server systems, different agencies or even transfer to notebooks can be modelled.

- Building a powerful base for visualization of areas of all kinds. This is especially of interest for „net surfer". To visualize a journey through the CyberSpace it is a very good metapher to show the way as a sequence of rooms linked together with doors (or gates). The space classes provide a good point to link this visulization information to and to provide a substantial semantic to the pictures.

- Probably the most important feature of the classes is the provision of very general and easy to comprehend abstractions for a lot of different entities and operations. Therefore it is possible to build in conjunction with the COA a set of generic and dynamic data manipulation tools. An example of this is an editor similar to a file manager, which may be used to manage and organize entities in and between different areas.

- And last not least these classes serve as an effective means to separate various aspects in engineering software systems, allowing to concentrate on the respective relevant aspects.

5 Conclusion

Defining the CyberSpace appropriate, the spatial aspects of software systems can be modelled in a very useful fashion.

An easy to survey but far reaching object oriented model is presented here, composed of the classes `Area`, `Frame`, `Context` and `Gate`.

This model can serve as a platform to build a great variety of applications on, allowing to access and describe many aspects of software systems in a clear and powerful manner.