## Circle Language Spec Plan System Objects Spec 2008-07 Project Summary

*Author: JJ van Zon*

*Location: Oosterhout, The Netherlands*

### Super-Project

Circle Language Spec, Command as a Concept Spec

### Goal

The System Objects articles were worked out,

to later build upon when working out the Assignment articles,

to later build upon when working out the Commands articles.

### Date & Time

July 14, 2008 – July 24, 2008

11 days

32 hours of work

### Products

The following was produced:

*System Objects article group*

version *2008-07-24 00 2.0*

**38** articles:

*System Objects*

*System Objects in a Diagram*

*Objects Floating Around*

*Objects Floating Around in a Diagram*

*Related Item*

*Related Item in a Diagram*

*Controlling a Related Object*

*Controlling a Related Object in a Diagram*

*Pointer To Pointer*

*Pointer To Pointer in a Diagram*

*Symbol*

*Symbol in a Diagram*

*Object Get & Set*

*Object Get & Set in a Diagram*

*Pointer To Pointer Get & Set*

*Pointer To Pointer Get & Set in a Diagram*

*Value Get & Set*

*Value Get & Set in a Diagram*

*New & Annul*

*New & Annul in a Diagram*

*Related Lists*

*Related Lists in a Diagram*

*Related List Item*

*Related List Item in a Diagram*

*Add*

*Add in a Diagram*

*Remove*

*Remove in a Diagram*

*Names of Related Items & Related Lists*

*Names of Related Items & Related Lists in a Diagram*

*Calling a System Command*

*Calling a System Command in a Diagram*

*Related Items & Related Lists Collections*

*Related Items & Related Lists Collections in a Diagram*

*Extending the System Interface*

*Extending the System Interface in a Diagram*

*System Objects Summary*

*System Objects Summary in a Diagram*

### Project Steps

- Created an article for each term

- Read over articles

- Reformulated articles

- Created diagram notation articles

- Covered details

### Reflection

This proved to be quite involved to make.

Would Circle run stand-alone as a run-time, the implementation of the *system objects* might cover most of the base of the code, upon which the rest could build, is the assumption. If Circle is just used as a diagram notation, system objects might not be as essential.

The description of the system objects might be needed for understanding other concepts, such as assignment, static members and aspects (from aspect object programming).

Sometimes progress seemed slower, by first doing the conceptual explanation and after that do all the diagram notation articles. When working on the diagram notations, it felt like not moving forward, because of the feeling you already did all this, with the conceptual explanation.

But that seemed the way to do it for the System Objects articles, because the conceptual explanation was created from a brainstorm, that was later split up into topics. Then pieces of text were dropped into the articles, and after that, the text was reformulated. And then, the diagram notation had yet to be come up with.

### Notes during the Project

#### Functional Aspects

##### System Objects

- Renamed the whole concept of System Interface to System Objects.

- Adapted the articles:

- System Objects

- Objects Floating Around

- Related Item

/ Controlling a Related Object

- Pointer to Pointer

/ Symbol

- Object Get & Set

- Pointer To Pointer Get & Set

- Value Get & Set

- New & Annul

- Related List

- Related List Item

/ Add

/ Remove

- Extending The System Interface

- Summary

- The term *system object* was defined somewhere, as a term mainly used for objects such as related items, related lists and related list items.

- The system objects and commands could be the basic objects and commands implemented by the base of the code. The system objects might not be 'normal' objects. They might be special deep-core system objects, that control relations between normal objects.  
In the first layer of the base of the code, the system objects and system commands might not be comparable to normal objects and commands in the new computer language. But the idea was that the base of the code would get reprogrammed using the new computer language itself and then the system objects and commands would be implemented the same as any plain old object or command. But they would *still* be *system* objects, would that have the special position of controlling the other objects.

- You might want to make clear, that the notation of the system objects and the implementation of system objects are two separate things.

- System objects such as Object . ID could also be considered system objects: they kind of are the related *objects*: an object reference embedded inside a related item or related list item. Perhaps put the term related object in the system interface summary, in the diagram that shows all the terms.

- Perhaps explain somewhere, why a system object is often called a system interface

The system object in a way represents an object. So usually you would see a system object as being the object itself, while the system object would really only control a pointer to the actual object. So when you see the members of target object of a system object, showing the actual members of the system object, it is like showing the *system interface* of a related object. That is why perhaps instead of calling it a system object, it might be called the system interface of a related object.

##### Related List Item Extension Procedures

- The notation for the list item extension procedures was not determined yet.

- After some drawing versions, the final drawing became was the notation to go for, for now.

- 2008-07-20 07

- 2008-07-21 01

> Drew it out differently: giving related list items a reference to a list, and give the list a reference to a system procedure.

- 2008-07-21 02

- 2008-07-21 03

- 2008-07-21 04

- 2008-07-21 05

Notation of extending a system command of a related list item.

- 2008-07-21 06

Notation of all extensible parts shown in the system interface of a related list.

- Decided to not draw out parameter passing

> This part of the documentation might not yet describe how to pass parameters, so you might not want to draw out parameter passing yet. Maybe mention that in the documentation, that parameter passing is not drawn out, because parameter passing will only be introduced later.

- This might just cover the notation, maybe not the way the system objects are tied together internally.

- Related List Item extension procedures altered the following articles:

- Related Lists

Added the related list item command extension objects to the sum-up of system interface members.

- Related List Item

Mainly added a small part about how the implementation is not covered there.

- Extended the System Interface

The Item commands and Gets and Sets in the system interface of the related list do not seem to be commands, but objects, with references to the extension procedures, because the Item Gets and Sets and commands in the related list do not seem to be executable commands, but *configurations* of the commands of the related list items, and this configuration would be stored by an object.

- Extended the System Interface in a Diagram

Showed how the extension of item elements of the system interface of related list looks.

- System Interface Summary

Added the related list item command extension objects to the system interface of a related list.

- System Interface Summary in a Diagram

Showed the diagram with all the extension objects of a related list.

##### Extension Procedure Part of Parent

- Extension procedures might not be defined by the list object. The extension procedures might be defined by the parent object. This counts for related items too. A related item’s extension procedures might be defined by the parent object. Extension procedures might be normal commands, not system commands.

> It seems just more practical to make the extension procedures part of the parent object. Only a normal Circle language object might define commands, that have all the possibilities of the programming language. You may not really be able to change objects like related items or related lists, because they are system objects. Maybe those system objects can only be extended, not changed.

##### Object References Inherently Part of Parent

> Theoretically it might not matter whether extension procedures are defined for instance inside a related item or inside the parent object of a related item. A related item might be part of the parent object anyway, as opposed to the actual object the related item refers to. In that case the reference to the object *is* part of the parent object. The object would not be.

> Objects referred to by a parent object might not be part of the parent object, but in that case the *references* to the object *would* bepart of the parent object. Objects such as a related list or a related item could be system objects that are inherently part of the parent object. Only the objects referred to would *not* be part of the parent in that case.

#### Writing Style

- In the article Related Item in a Diagram, the complete system interface of the related item was not shown, even though you might expect a full overview there. You only were redirected to other articles for that. The reason was, because the system commands have an explanation in the other articles. But an overview might still be nice, even when not everything is explained yet.

> Related List Item

- Maybe the summary article is a bit incomplete, because not everything seems to be mentioned in it.

#### Other Issues

- In the member articles (e.g. Object Get & Set) perhaps refer to the articles about the system objects to which the members apply.

- You may want a parent object to be able to define multiple extension procedures for the same kind of extension, for the same related item or related list, because when you let an object support an aspect, the aspect can add another extension to a the system command. So an extension procedure might not be a single-cast event.

- Access control

A lot of things about the system interface could be access controlled. Access control might only be worked out in the future. You might however mention it in the documentation in multiple places, where access control could be applied. Maybe refer to the access control documentation, but also maybe mention the access control’s physical effect.

An idea for access control documentation, might be that you could consider the ability to access control disparate little aspects of the system interface.

- Notation of call to a system command.

- Consider related item’s ID In Parent…

> Right now (2008-07-21 19:16) the idea is, maybe this is part of relationship synchronization.

> Maybe not (2008-07-24 19:27). It might be part of the concept of having a list of Related Items inside the parent object.

- System objects / base of the code / aspects,

2008-07-24

System Objects might cover the complete base of the code.

This means, that for a complete overview, you would still need the Related Items and Related Lists system objects. But the usage of system objects may not require knowledge of the objects Related Items and Related Lists. However, in order to author aspects, you might want those collections available. You also may need to be able to extract an Attributes collection as well. And a methods collection.

It is worth considering making the story complete like that, so the whole base of the code might be covered that way.

But it already has been quite some work, and maybe just get on with it for now.

- Related items and related lists have names. The name might be a system member, because it may needs to be tapped into by aspects.

- Maybe add these members to summaries and the articles related item and related list:

- ID In Parent

- Name

- Maybe add something about the collections Related Items and Related Lists to the summary.

- Any references to the main article are missing. There are no redirections from the main article to the sub-articles. But it is not sure, if that's a problem.

- Maybe add references to the articles of the members of a system interface to the articles about for instance Related Item, Related List and Related List Item.

> Anything in brainstorm texts is probably worth mentioning somewhere in the documentation, because if the initial writer had trouble with it, it is probably worth explaining to somebody else, that might be dealing with the same material later.