



# Glueing Components in Wide Area Networks

---

Werner Van Belle

Official Member of *Quick & Dirty (inc)*



# Glueing Components Together

---

- Doesn't work in Distributed Environments
  - Need Stub – compilers (non dynamic)
  - Changing Implementations
  - Very Static Linkage between Components



# Glueing Components Together

---

- Advantages in distributed environments
  - Rigid Defined Protocol between components (contrary to OO)
  - Real Data Encapsulation
  - Loosely coupled



# Glueing Components Together

---

- Writing distributed applications
  - Is hard, because we have to implement the protocol ourselves
  - Is difficult, because it's almost always asynchronously
  - Is bothersome, because we have to take errors into account
  - Makes you tired, because you have to implement a new MOP each time you communicate with remote objects.



## But the ...

---

- Real problem lies in the language constructs which are offered.
  - They only aim at synchronized communication.
  - They enforce a certain calling methodology upon the programmer



# Call-With-Current-Continuation

---

- Problems:
    - Difficult to explain
    - Difficult to understand
    - Difficult to use
- Badly Integrated into current day languages



# The Return Continuation

---

- Makes things easier
  - To explain: The return continuation represents what will happen when your function returns.
  - Easy to use:



## A typical *Q & D.irty (Inc)* example (1)

---

```
Ctx: void  
CalculateAsync()  
  {Ctx:=return;  
  void}
```

```
{display(CalculateAsync());  
display("test")}
```

```
:: voidtest
```





## A typical *Q* & *D.irty (Inc)* example (2)

---

Ctx(100)  
:: 100test

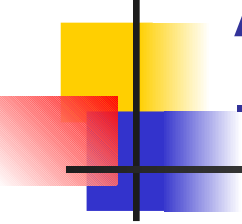
Ctx(5)  
:: 5test



# Furthermore: The Return Continuation

---

- Is definitely ‘more cool’
  - The return continuation can be called directly from within a function.
  - The return continuation can be send a message to as a form of exception handling



## Another typical *Q* & *D* example -- calling the return continuation

---

```
somePicoFunction(t)::  
  {  
    if(is_void(t), return(0), false);  
    if(is_text(t), return(1), false);  
    if(is_number(t),  
return(void),false);  
    2  
  }
```



## Another typical *Q* & *D* example -- sending messages to the return

---

```
Notatable()::  
  display("sorry...");  
  
somePicoFunction(t)::  
  if(not(is_table(t)),  
      return.Notatable(),  
      2);  
  
display(somePicoFunction(30))  
:: sorry...
```



# Still Further More:

---

- The return continuation allows
  - The implementation of the Arrow-operator: a way to happiness in distributed environments



# The Arrow Operator

---

- Changes the return of the receiver

`a..calculate(50)->display`

- Allows a redefinition of the Standard Control Flow
- Will be implemented in the next release of Cborg called: *Borg on Cubes*.