In Pursuit

of

The

Grand

Unification

Theory of

Computer Operation

"True Innovation Always Bucks The System"

The *Gadgetorium*, LLC
6430 N. Michigan Road
Indianapolis, IN 46268
Voice: (317) 251-3546
Email: tlh@gadgetorium.com

CONFIDENTIALITY AND NONDISCLOSURE AGREEMENT

WHEREAS, ***The Gadgetorium, LLC*** agrees to furnish **You** (_____) certain confidential information relating to ideas, inventions or products for the purposes of determining an interest in developing, manufacturing, selling and/or joint venturing;

WHEREAS, **You** agree to review, examine, inspect or obtain such confidential information only for the purposes described above, and to otherwise hold such information confidential pursuant to the terms of this Agreement.

BE IT KNOWN, that ***The Gadgetorium, LLC*** has or shall furnish to **You** certain confidential information and may further allow **You** the right to discuss or interview representatives of ***The Gadgetorium, LLC*** on the following conditions:

1. **You** agree to hold confidential or proprietary information or trade secrets ("confidential information") in trust and confidence and agrees that it shall be used only for the contemplated purposes, shall not be used for any other purpose, or disclosed to any third party.
2. No copies will be made or retained of any written information or prototypes supplied without the permission of ***The Gadgetorium, LLC***.
3. At the conclusion of any discussions, or upon demand by ***The Gadgetorium, LLC***, all confidential information, including prototypes, written notes, photographs, sketches, models, memoranda or notes taken shall be returned to ***The Gadgetorium, LLC***.
4. Confidential information shall not be disclosed to any employee, consultant or third party unless they agree to execute and be bound by the terms of this Agreement, and have been approved by ***The Gadgetorium, LLC***.
5. This Agreement and its validity, construction and effect shall be governed by the laws of the United States of America.

AGREED AND ACCEPTED BY:

Date:_____

By_____ Witness:_____

Title:_____

By_____

Title_____

An Introduction to the Gadget-Based Computer Programming System

The *Gadgetorium*, LLC
6430 N. Michigan Road
Indianapolis, IN 46268
Voice: (317) 251-3546

Introduction:

New paradigms in computer functionality are inevitable. We will see these changes in many areas. Programming concepts and methods, data representation and human interfaces will all evolve. These statements assume that the way it is done now is not as it should be yet. If it were already perfect, change would be a detriment.

This is where this technology comes into play. It represents many years of work devoted to the furtherance of computer efficiency. It can and should be taken to the "next level".

Here is how this can be done:

Step 1: Define a Universally Applicable, Standardized Data Representation Scheme

Step 2: Modularize Programming by Defining All Program Methods within the Data Representation Scheme

Step 3: Eliminate Database Structure Programming at the Application Level

Step 4: Allow Applications to Access (Remotely Control) Other Application's Methods Easily and Efficiently

Step 5: Create Tools Using These Concepts to Allow Rapid Development of Vertical Applications

Step 6: Provide a Natural Language (English) Interface to the Data

Step 7: Provide a Voice Interface to the Data

Step 8: Commercialize the product

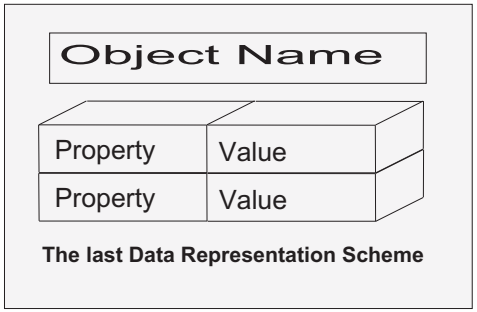
See the following pages for more details about the concepts and systems based on them.

Evolving Software from a Series of Mistakes (OOPS)
To A True Object Programming Scenario (TOPS)

The Gadget Concept

The Gadget Philosophy requires the acceptance of one self evident truth about programming. It is up to you to prove to yourself that this statement is correct. The more, the better.

"All Data should be represented as shown at right"



The last Data Representation Scheme

Representing data becomes an exercise in answering the following question for all objects.
What are the Properties of the (fill in the blank) Object?

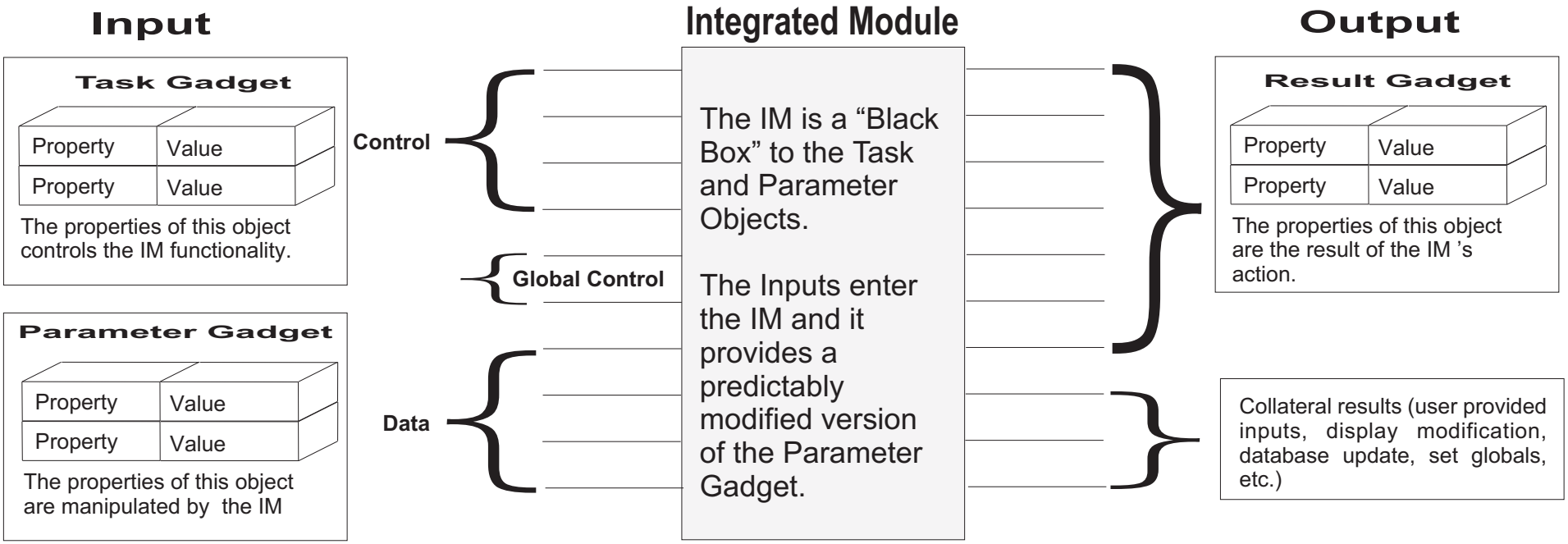
By representing all data in a software system as a Gadget (an Object), a whole world of new possibilities opens. The increased modularity in itself is a massive time saver. If I, as a user, can depend on every data element in a system to have a consistent structure, I can virtually forget the details about how to recall, memorize and forget them. I can now devote my efforts to the changing of the properties of the elements to reflect any new needs of my particular situation.

For instance, if I am developing an application to modify customer data in some way, all I need to know is what properties a customer has, not how a customer is stored. I do not need to be a mechanic. I want to drive the car, not build it! This extends from programmer to end user and thereby puts both on the same page, so to speak.

This document introduces a number of things based on this simple philosophy. The Integrated Module (IM) or Software Chip idea is presented as a standardized programming construct. Next, a description of a modular programming environment called The Virtual Cortex System which implements this technique. You will also see an interesting use of this concept to create an automated distributed application scenario via DDE communications between applications.

Step 1: Define a Universally Applicable, Standardized Data Representation Scheme (The Gadget)

Borrowing from the electronics industry, we can envision a "Software Chip" or "Integrated Module" (IM) as the software equivalent to an Integrated Circuit (IC). By doing this we can visualize software operations more effectively.



See a pattern here? Standardizing Data Representation relieves a large portion of miscommunication between all parties.

Programmers and users describing the same thing the same way, What a concept..

Integrated Module pseudocode Description
Important Note: The IM IS A GADGET!

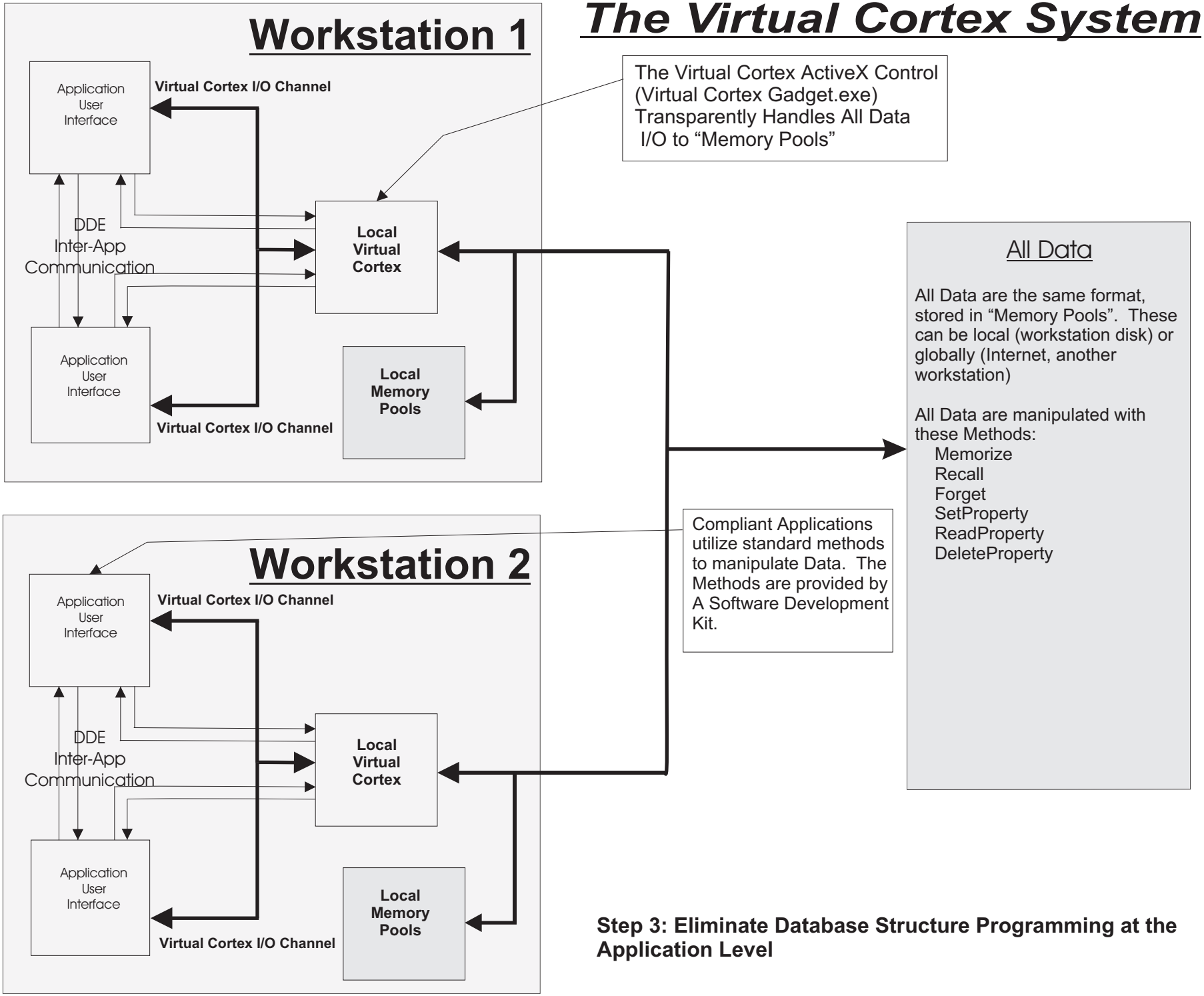
All Gadget Compliant Integrated Modules Have the Following Structure:

```
Function imFunctionName (Task as Gadget, Parameter as Gadget) as Gadget()
   The Task Object contains the parameters of the function
   The Parameter contains the Gadget to work on.

      Do Whatever imMyFunction Does

      Select Case True
         Case ReadProperty (Task,"Method to Call")="imFunction2"
               Result() = imFunction2 (Task, Parameter)
      End Select
      Set Return Gadgets
      Go Away
End Function.
```

Armed with this tool, we can now begin to assemble a system that feeds on its ease of design for rapidly assembling universally compatible and complex application software like a ***Virtual Cortex.***

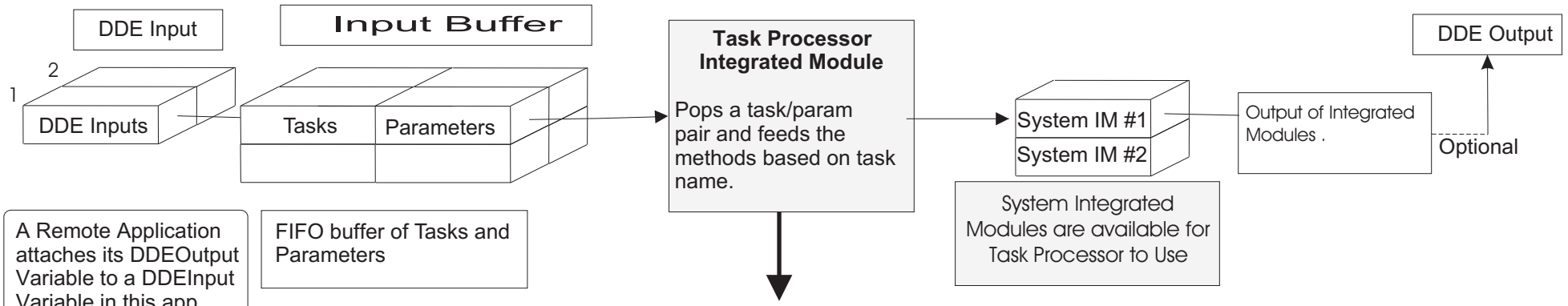
Step 2: Modularize Programming by Defining All Program Methods within the Data Representation Scheme

The Virtual Cortex System



Step 3: Eliminate Database Structure Programming at the Application Level

OPAG DDE Interface and Task Processing System Description*



A Remote Application attaches its DDEOutput Variable to a DDEInput Variable in this app.

DDEInput On Change Events push the contents onto the Task Processor Stack for automatic handling.

The Task Processor pops the Task and Parameter Gadgets and calls the correct method.
See description.

Note: The Remote Program uses the ***Dematerialize*** function to transfer Gadgets as text. There must be 2 On Change Events to transfer the complete task/parameter pair.

FIFO buffer of Tasks and Parameters

Task Processor Integrated Module Description in VB Pseudocode (SDK Module: PUNITS.BAS)

Note: the im prefix designates an integrated module method.

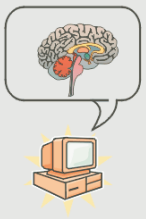
```
Function imTPU (Task as Gadget, Parameter as Gadget) as Gadget
   ' The Task Processing Unit is a Case Statement with the names and parameters for every Function
   ' available for remote control by this and other applications. These will be maintained as Gadget
   ' Integrated Module (GIM) Libraries which are imported from the Vbasic Project Files.  A market for
   ' GIM Libraries will stretch into the millions if done correctly.

   Select Case True
      Case ReadProperty (Task,"Method to Call")="imMyFunction"
            Result = imMyFunction (Task, Parameter)
            SetProperty Result, "Some Status Stuff","OK"
   End Select
   'Do Post Processing Stuff Here
   'Bye You all
End Function
```

All Compliant Functions (Integrated Modules) Have the Following Structure:

```
Function imMyFunction (Task as Gadget, Parameter as Gadget) as Gadget()
   The Task Object contains the parameters of the function
   The Parameter contains the Gadget to work on.
      Do Whatever MyFunction Does
      Set Return Gadgets
      Go Away
End Function.
```

* Object Processing Application Gadgets (OPAGs) are Software Applications that conform to the concepts presented in this document

Step 4: Allow Applications to Access (Remotely Control) Other Application's Methods Easily and Efficiently



Features of The Virtual Cortex Gadget Software Development Kit (SDK)

Transparent Database I/O
- Eliminates Data Structure Manipulation
- Enables programmers to be more efficient

A Set of ActiveX Controls Provide Gadget Aware Functionality
- Gadget Aware List Control
- Gadget Aware Combo Box Control

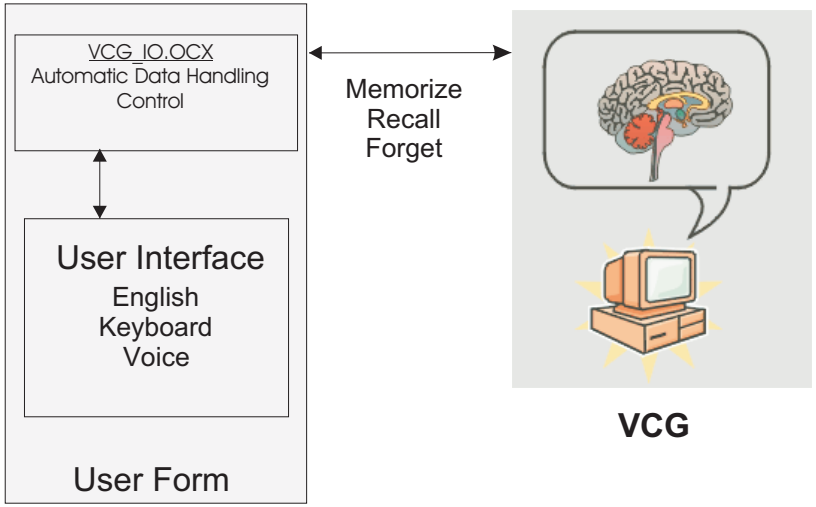
English-based Data Interface Manages Data More Naturally
- Make A Person have Name, Address, City, State, Zip, Weight, Height
- Make John Doe be A Person (John Doe is A Person works too)
- Is John Doe A Person?
- Make John Doe's height=5 ft 11 in
- What is height of John Doe? (What is John Doe's height?)
- Recall Names with: Color = Red
- Recall Names with: Color, Length
- Recall Values of Color
- What has color?

Speech Interface (requires optional, third party speech engines)
- Voice Recognition Interface to the English-based Data Interface
- Includes Synthesized Speech Output

Intelligent Gadget Integrated Module Development Wizard
- Extract Methods from Source files into *GIM*s (VB only now)
- Automatically Create *GIM*s from Gadgets with Drag/Drop
- *GIM* Library Manager Encourages Programmer Cooperation
- Build Applications by Drag/Drop Module Manipulation

Visual Gadget Creation Tools
- Gadget Manager Creates Reusable Objects
- Drag and Drop Object Configuration

Fully Configurable Data Recall and Display via Filters
- Recall Filters Recall Data the Way You Define
- Display Filters Present Data the Way You Define



Step 5: Create Tools Using These Concepts to Allow Rapid Development of Vertical Applications

Notes on the Interactive GIM Wizard:

Create a new Gadget Integrated Module (a VB Function that complies to the previous description) by analyzing the properties of the dropped object and creating if then statements based on the properties of the dropee.

Example:

Customer has properties Name, Address, City

The IM would be:

```
Function imDoCustomer (WhatToDo as Gadget, Customer as Gadget) as Gadget
' WhatToDo is the Task Gadget, Customer is the Parameter Gadget
    If ReadProperty (Customer, "Name") = "Joe Blow" then
        ' Do Whatever you need to do
    End If

    If ReadProperty ((WhatToDo, "Look up Address") = "True" then
        ' Do Whatever you need to do
    End If
    etc...
    imDoCustomer = Result
End Function
```

Example, in American English:

Create a template for something, a person:
Make A Person have Name, Address, City, State, Zip and Phone
This creates a "person" with the defined attributes

Now, every time you want to make a memory the same as the template just do this:
Make "Joe Blow" be "a person"

Modify the properties of a memory like this:
Make "Joe Blow" have Name = Joe Blow, Address = 123 Somewhere St., City = Any town, State = Any State, Zip = 12345, Phone=123.456.7890 in My Friends

Note: If Memory Pool "My Friends" doesn't exist, it is created.

Recalling Computer Memories:

Recall Joe Blow from my friends

Recall Names with: zip=12345

What is Joe Blow's Address, City, State and Zip?

What has Address?

What has Address = 123 Somewhere St.?

Forgetting Computer Memories:

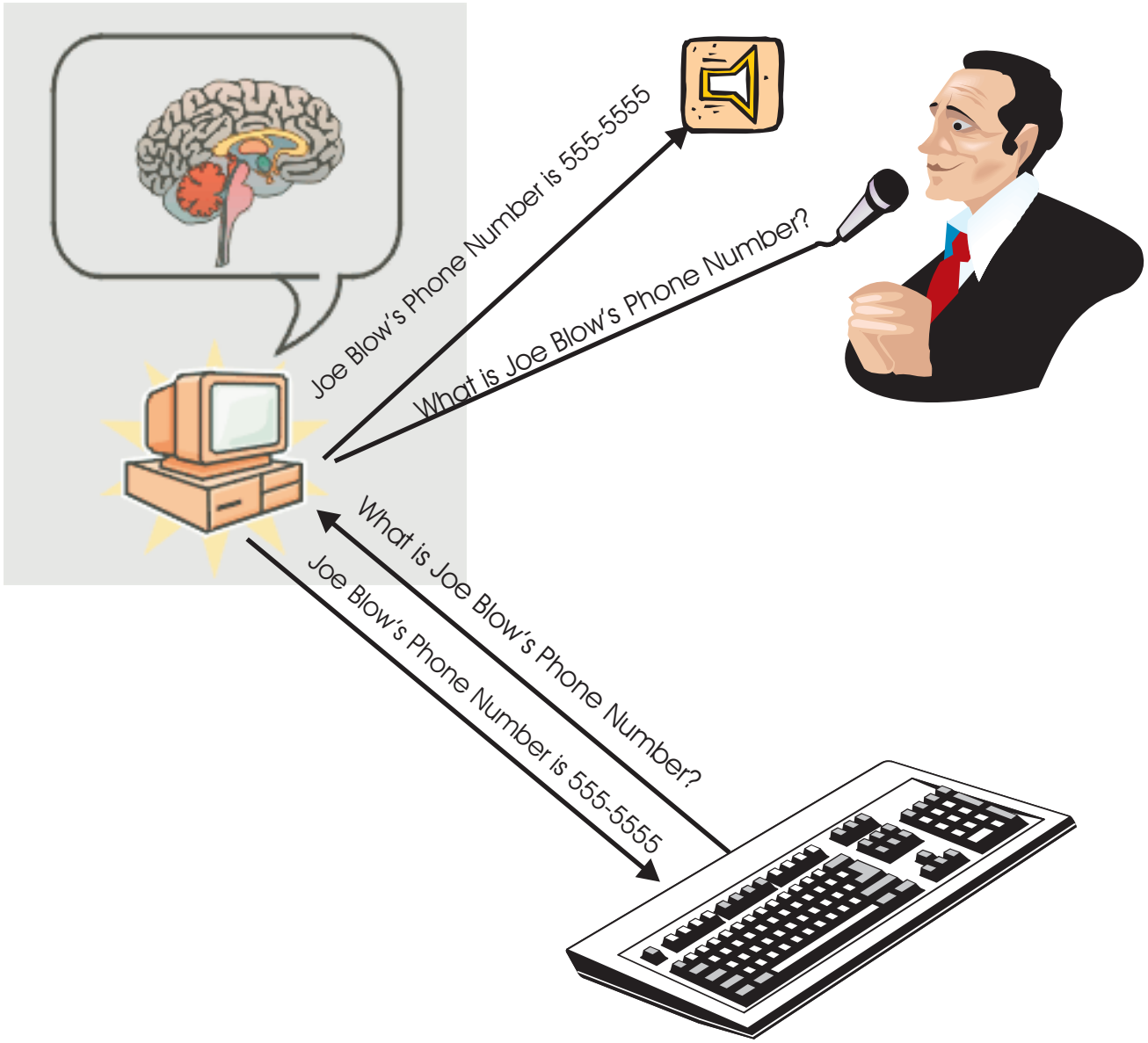
Forget Joe Blow (forgets the memory and all properties).
Forget Joe Blow's City (forgets an individual property).

Examples of the concept:

Pixel
X=12
Y=34
Color=Blue
Intensity=4

DNA-Gene Locus 45-33-b43
Base 1=Uracil
Base 2=Tyrosine
Base 100000=Whatever Base 100000 is

Step 6: Provide a Natural Language (English) Interface to the Data



Step 7: Provide a Voice Interface to the Data

Where Do We Go From Here?

We have a number of the parts needed for successful commercialization of the concepts. How do we proceed from here?

- Get the resources to finish:
  - The VCG Needs
    - Updated to handle a number of database formats (pure SQL)
    - Enhance the file import functions
    - Converted to the IM concept (it was written before the IM was solidified)
  - The Controls Need:
    - Gadget Aware List Control
      - Finish Interface
      - Test
    - Gadget Aware Combo Box Control
      - Same as The List Control

Step 8: Commercialize the product