# RAD Design

## *UI*

### UI Paradigm

### Modeled From Outlook

We think outlook is a great example client for RAD.  Not only would a UI modeled after it be very familiar to the user, but a lot of studies and a lot of years of UI development has already went into Outlook.  We just glean from its wisdom.

### Tree View – List View – Detail View

Everything drills down from a tree view, to a list view, to a detail view.  At the top level, the tree view one is dealing with collections of grouping, each grouping driving a business need.  The next level, the list view, one is comparing, observing relations or locating.  At the last level, the detail view one is performing actions or changing values of a single object.

#### *Tree View*

Each node in the tree view is a group or collection of identical object types.  These object types are things like candidates, jobs, placements, or candidate job matching.

The purpose of each node is not to organize objects base on relationship, rather to organize based on workflow.  For example, inside a node containing jobs one would not see sub nodes representing each candidates, or sub nodes representing each recruiter because such a list is based on relationship.  Relationship organization is handled by the list view.  Within the jobs node one would see sub nodes such as "new jobs today", "jobs coming to a close" and "jobs needing interviews".

The described behaviors of the tree view nodes classify the nodes as a "bucket".  Each node represents a bucket of objects.  Each bucket has a criteria associated to it.  Each object in a bucket in there because it meets the buckets criteria.  For example, the bucket "my new jobs today" has the criteria of any job assigned to the recruiter that has been entered today.  Therefore, only jobs meeting that criteria are in the bucket.

There are calculated buckets and user defined buckets.  A calculated bucket, like "my new jobs today" is maintained and kept up to date by RAD.  There is no way for a user to directly drop a job into such a bucket.  A calculated bucket is global to all users and should suggest a workflow by nature.  A user defined bucket, like "SDET" is specific to the user and only appears on there profile.  Furthermore, The objects in the bucket are maintained by the user and not the system.  Since each type of bucket works differently there will be a visual indication to identify calculated and user defined buckets.  For example, a calculated bucket could be in italics.

The same object can be in multiple buckets.  Both user defined and calculated.  For example, the same candidate could be in the "new candidates today" bucket and the "unqualified" bucket.

### *List View*

Since the items in the list view are derived from a bucket, the list view only ever contains a list of one type of objects at a given time. For example, the list view would not be showing a mixture of candidates and a mixture of jobs, either it contains one or the other. For each type of object in the list are associated a collection of defrent views that can be applied to the list. Applying different views to a list is what enables a user to change how they compare objects and observe relations.

A "view" is a stored single entity consists of a collection of columns, sort orders, group bys and filters. Group bys will create hierarchy in the list and add expandable collapsible widgets to the list.

To make views a natural part of a list view they will be intuitive to configure, like Outlook. But unlike Outlook, they will also be easy to save, reuse and switch from one to the other.

Some initial views will be preconfigured for each users, but all views will be user defined. Meaning that if one user adds new views or modified existing views it only effects that users profile.

### *Detail View*

The two types of detail view for each type of object are preview pane detail and popup window detail. A preview pane, like Outlook, will be available to show the details of a selected object from the list view. For very complex object types like a candidate or a job, the preview pane detail will be summarized and possibly read only, and in no way reflect everything about the object. For full access to the features of a complex object type, the object will have to be double click from the list view, opening the single object in its own window. For simpler object types, like a submittal, the preview pane detail and the popup window detail would be the same.

The features of the popup window detail and the not summarized preview pane detail are available actions, suggestive actions, view object values, change object values, view relations, access relations.

I think this still needs a lot of thought and ideas to figure out how move the detail view from simply a bloated conglomerate of being able to edit everything and being able to see a hundred relations, like EZ, to something that moves the recruiter through a structured workflow.

## How RAD fits into the UI Paradigm

## Layout

The same way Outlook currently looks. Including a favorites list in the upper left. This becomes a collection of high visibility buckets. These are bucket you not only want to have quick access to but want to see when they change. A bucket in this window might create a notification when something is added to the bucket. For example, if ones bucket in the high visibility window was "my new job orders" then RAD might notify the user the moment a new job order is received. This could be done the same way Outlook notifies of a new email item.

The other change from outlook is that we would not need the buttons on the left bottom of Outlook. Outlook uses these to move from one module to another; from mail to calendar.  Since our layout will be able to fit everything into the tree view its not necessary to change modules.

## Example of the tree view and buckets

Below is an example of what a recruiters tree view might look like as we thought through fitting in all the functionality from RAD.  The italic items represent calculated buckets, the bold ones represent module organizations and the normal ones represent user defined buckets.

- **Candidates**
    - My Candidates
        - Developers
        - Testers
        - Project Managers
        - Truck Drivers
        - *Shared Candidates*
        - *Candidates Last Contacted in 6 months*
    - *Candidates I've qualified that need to be contacted soon*
    - *Candidates I'm trying to contact*
    - *Everyone's Shared Candidates*
    - *Redeploy Candidates*
- **Jobs**
    - *My Jobs*
        - Tester Jobs
        - Project Management Jobs
        - *Jobs with interviews*
        - *Jobs with submittals*
    - *Placements this month*
    - *All Active Jobs*

- **Matches**
    - *Candidate Searching*
        - Save Query from 1/1/2004
        - Save Query from 10/6/2004
    - *Fuzzy Searching*
        - Save Query from 1/1/2004
        - Save Query from 10/6/2004
    - *Fuzzy Matching*
        - Save Query from 1/1/2004
        - *AutoMatched to my jobs*
- **Hiring**
    - **Submittals**
        - *Pending feedback*
    - **Interviews**
        - *Upcoming interviews*
        - *Pending feedback*
    - Placements

## UI details

## Real Buckets

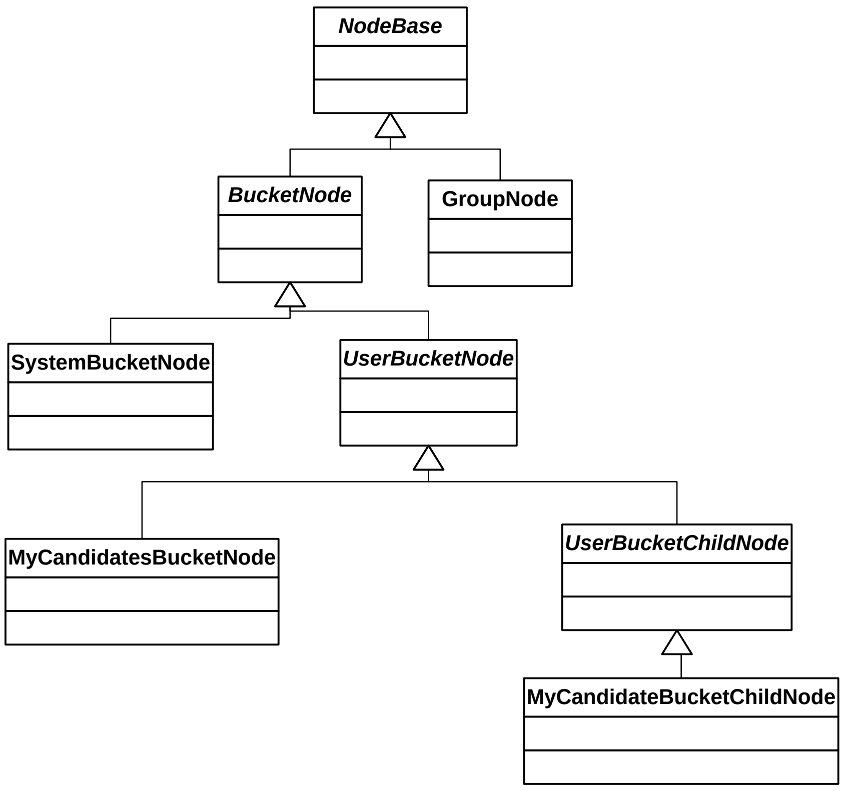
- Candidates
    - My Candidates
        - <user defined buckets>
        - Sourcing
            - Contacting –Contacting
            - Qualified - Qualified
            - Pending Background Check- ????
            - Expired Contacts -ContactExpired
            - Expired Qualifies - QualifiedExpired
        - Submitted- Submitted
        - Interviewed - Interviewed
        - Placed - Placed
        - Candidates needing contact - NeedingContact
    - Shared Candidates
        - <shared user defined buckets>
    - Expired Contacts - ContactExpired
    - Expired Qualifies - QualifiedExpired
    - New Candidates (last 3 months) - New
    - Newly Qualified Candidates - QualifiedNew
    - New Hire Board (this month) - PlacementsNew
    - Placements (1 year) - PlacementsNew
    - Redeploys - Redeploy
    - MS on 100 day break - MSBreak
    - Current Employees - ExcellEmployee
- Reqs
    - My Reqs - All
        - <user defined buckets>
        - Open - Open
        - Closed - Closed
        - Needs Submittals - NeedSubmittals
        - Needs Interviews - NeedInterviews
        - Placed - Placed
    - Shared Reqs
        - <shared user defined buckets> (one will be “Hot Jobs”)
    - Open - Open
- Managers
    - My Managers
        - <user defined buckets>
        - Managers with open job orders - OpenJobs
        - Managers needing contact  - NeedingContact
    - Managers who submit lots of Reqs -FrequentJobSubmittal
- Submittals
    - My Submittals

    - Submittals against open Reqs - OpenJobs
    - New Submittals - New
  - New Submittals - New
- Interviews
  - My Interviews
    - Interviews against open Reqs - OpenJobs
    - Upcoming Interviews -Upcoming
    - New Interviews - New
  - Upcoming Interviews - Upcoming
  - New Interviews - New
- Placements
  - My Placements -All
  - Recent Placements (1 year) - New
- Staying in Touch
  - Contacting – Contacting (candidate / with userID)
  - Qualified – Qualified (candidate / with userID)
  - My Expired Contacts – ContactExpired (candidate / with userID)
  - My Expired Qualifies – QualifiedExpired (candidate / with userID)
  - Managers needing contact – NeedingContact (manager / with userID)
  - Candidates needing contact – NeedingContact (candidate / with userID)
- Finding Candidates to Submit
  - Redeploys – Redeploy (candidate)
  - MS on 100 day break – MSBreak (candidate)
- Get Referrals
  - Managers who submit lots of Reqs – FrequentJobSubmittal (manager)
  - Current Employees – ExcellEmployee (candidate)

## Object Models

### *TreeView*

This figure shows the hierarchy of the nodes that compose the TreeView.



***NodeBase:*** *abstract* base class that provides common methods, properties and attributes used by the derived classes. The main functionalities provided are:

- Maintains a reference to the parent node.
- Maintains the collections of the child nodes and provides method to add a new child element.
- Maintains the reference to the control that represents the ListView for the current selected bucket.
- Defines whether the preview pane should be shown or not.

***Bucket:*** *abstract* class that defines the characteristics of a bucket node's type. The main functionalities provided are:

- Maintains the object type associated to the bucket.
- Maintains the collection of default and user defined views that specify the layout of the ListView associated to the bucket.

**SystemBucket:** is a specification of the Bucket type and defines system buckets that cannot be modified by the user.

**UserBucket and UserBucketChild:** is a specification of the Bucket type and defines buckets whose child elements can be added, removed or modified by the user. The main functionalities provided are:

- Add new child element.

- Remove an existing child element (UserBucketChild only).
- Rename an existing child element (UserBucketChild only).
- Defines hierarchies and sub-hierarchies of the child elements (like Outlook folders) (UserBucketChild only).
- Is possible modify the content of this child elements in a way that can maintains subsets of elements of type defined by the object type associated to the parent user bucket (for example a user defined bucket contains the categories that can be associated to a MyCandidate element. The user can drag and drop candidates element from the MyCandidates ListView to the node representing the category in order to associate this candidate to that category)( UserBucketChild only).

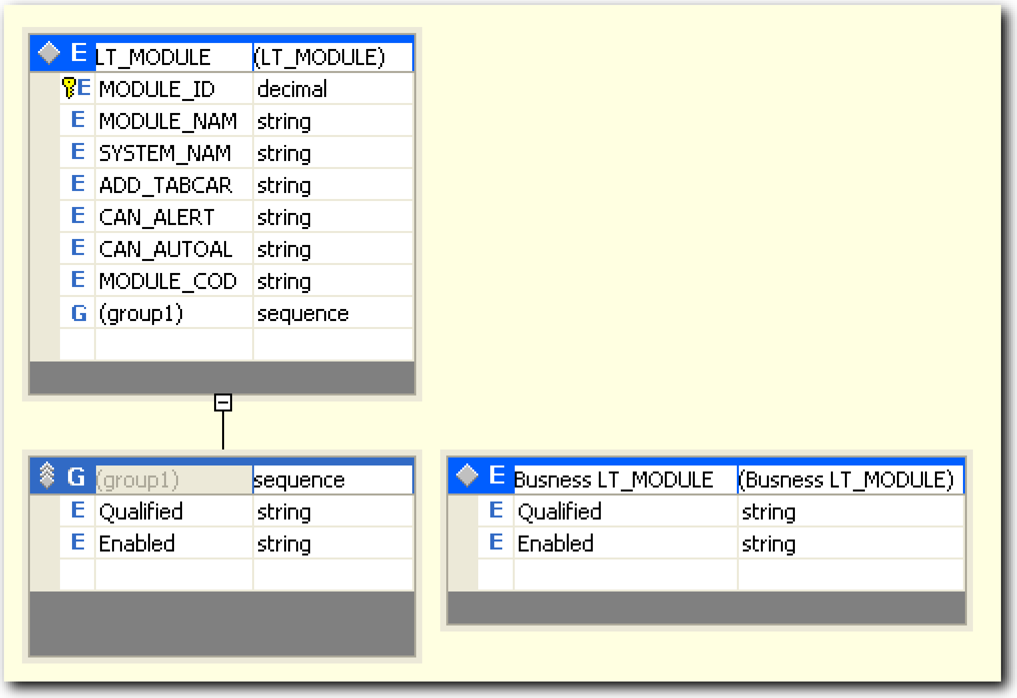
**GroupNode:** is used to group together nodes with a similar meaning. Nodes of this type can have associated ListView controls that show a summary of the contained nodes.

## *Business*

### Methodology

### Buisness Column in Datasets

Datasets are apart of the data layer.  All datasets will be an exact, unchanged, replication of what VS automaticly generates when the “Generate Dataset” action is executed in design time from a data adaptor.  However, there will be need of our business layer to have aditional data extended to our datasets.  For example, for a candidate row from the database, there will be a need to have business data associated to the candidate that indicates a calculated status for the candidate.  This aditional data must be associated to each row in our datasets and must be data bindable to our UI.  To acomplish this we will have extended columns in some datasets that reprecent Business columns.  We do not, however, want these columns to be overriten when we regenerate our datasets so we have established a means to keep both clearly seporated, yet still apart of the same dataset.  Each dataset that requires extended business columns will do so using a column in the dataset.  When the original dataset is regenerated the group information is overwritten.  To get around this problem, an exact replica of the group will be created as a seporate table in the same dataset.  This table will never be used for anything other than a means to store a copy of the business column group.  When the dataset is regenerated, the replica table can be used to copy and paste all the business columns back into the regenerated dataset in one simple action.  In this manor the datasets will visualy have a clear distinction between data layer data and busness layer data in the designer and will also provide a way to update the data layer columns without loosing changes to our busness layer columns.  The following image ilistrates this technique.



| | | |
|---|---|---|
| E | LT_MODULE | (LT_MODULE) |
| E | MODULE_ID | decimal |
| E | MODULE_NAM | string |
| E | SYSTEM_NAM | string |
| E | ADD_TABCAR | string |
| E | CAN_ALERT | string |
| E | CAN_AUTOAL | string |
| E | MODULE_COD | string |
| G | (group1) | sequence |

| | | |
|---|---|---|
| G | (group1) | sequence |
| E | Qualified | string |
| E | Enabled | string |

| | | |
|---|---|---|
| E | Busness LT_MODULE | (Busness LT_MODULE) |
| E | Qualified | string |
| E | Enabled | string |

In the ilistration, the data layer table is LT_MODULE, the business layer table is "Busness LT_MODULE" and is copyed into the data layer table as a group.

## Wrapped Datasets

Dataset that have special needs in managing a busness portion of the dataset will be wrapped in a business layer dataset class. These classes inherit from the original dataset class and override properties, methods and tap into events that allow managing the busness aspects of the dataset. These are things like calculating all the business layer columns, enforcing busness rules on data layer columns, or auto populating columns when a new row may be inserted.
The dataset class generated by VS for a strongly typed dataset does not allow overriding any of the properties and methods needed to wrap the dataset with a busness dataset class. Nor does is expose some of the events needed to trap things like row modifications. To get around this, the default custom tool "MSDataSetGenerator" that comes with VS is replaced on datasets needing the business wrapper with "AGDataSetGenerator". This generator will auto generate a class with all the things we need being over-rideable. The generator is supplied here.

ADONETPowerToysInstaller.exe

Also supplied is an example class with dataset and business wrapper.

Dataset1.xsd DataSet1BS.cs

## Naming Convention

## Object Models

## *Data Access*

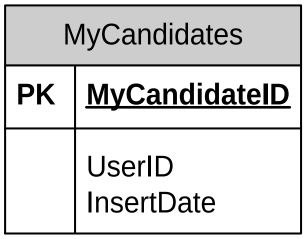
### Data Schema

#### *HotCandidates*

Candidates that are considered interesting to a specific recruiter or to the entire recruiting team are stored in this table. In order to *execute operations* (like submit, interview or placement) with a HotCandidate, this must be associate to one or more jobs.

*If a UserID is not specified, the HotCandidate is considered as Shared (visible by all recruiters).*

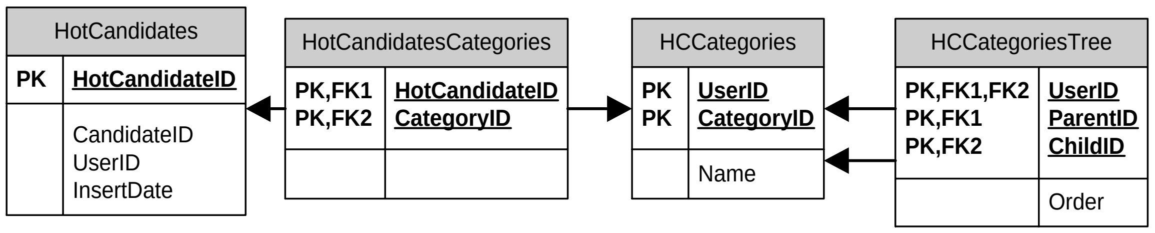
The following diagram shows the relationship between HotCandidates table and the job orders associated to a HotCandidate:



| MyCandidates | |
|---|---|
| **PK** | **MyCandidateID** |
| | UserID<br>InsertDate |

#### *MyCandidates Categories*

A recruiter can set up a *user defined hierarchy of categories* used to group together and categorize HotCandidate members. A HotCandidate can belong to multiple categories. The *HCCategoriesTree.Order* field is used to define the sorting criteria for the nodes and sub nodes of the hierarchy.



### *MyJobs Categories*

## Object Models