Molecular Genetics eXplorer

# Progress Presentation for Molecular Genetics eXplorer MGX 1.0

Department of Computer Science
University of Massachusetts, Boston
December 9th, 2004

## MGX Vision

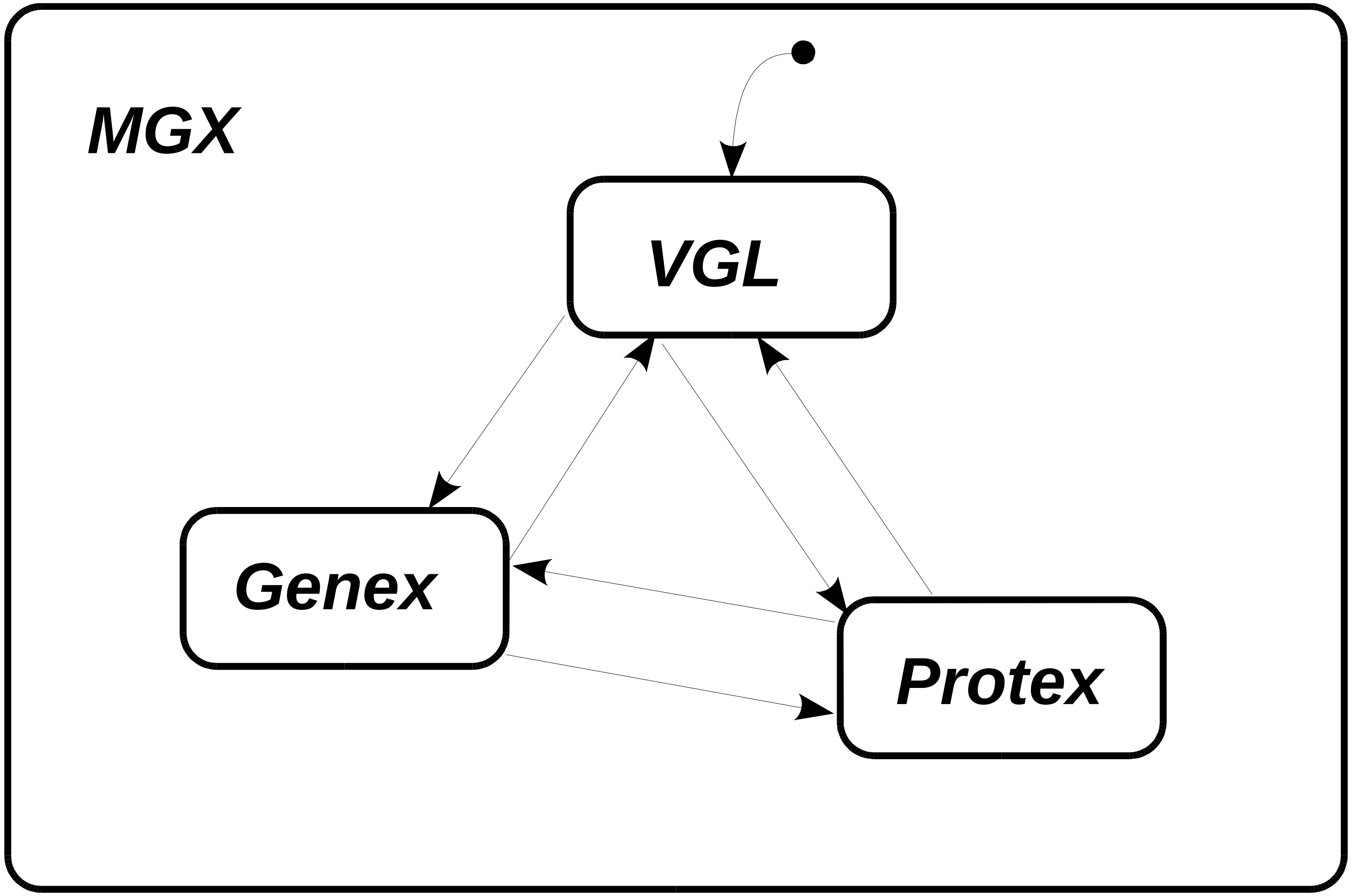
**Create a computer-based teaching tool that helps students to understand connections among Genetics, Molecular Biology and Biochemistry.**

## MGX Application (1)

- Three virtual biological laboratories.
  - VGL, a virtual genetics lab: Investigate the mechanism of inheritance for one trait.
  - Genex, a gene exploration lab: Transcribe and translate a DNA sequence.
  - Protex, a protein exploration lab: Visualize the structure and function of a protein.

## MGX Application (2)

- Two types of actors.
  - Student(s): General Biology 111/112.
  - Administrator: Professor Brian White.
- Two modes.
  - Integrated.
  - Stand-alone.



Statechart after Harel, D. 1987. Statecharts: A visual formalism for complex systems. Science of Computer Programming, **8**, 231-274.

## MGX: Development

**Bring together computer codes written over time by various teams and individuals.**

| ***Application*** | ***Written By*** | ***During*** |
|---|---|---|
| **VGL** | **CS students** | **2002-2003** |
| **GenExplorer** | **CS students** | **2003-2004** |
| **Genex** | **Prof. B. White** | **2004** |
| **Folding** | **Prof. E. Bolker** | **2004** |

## MGX: Existing Codes

- Which codes shall we take?
- How shall we make use of those codes?
    - For design.

      Use cases (David).
    - For implementation.

      One-step build (Pradeep).

## Use Cases: Definition

- A case of use.
- A narrative description of the interactions between a user and a system.
- An external or black-box view of functionality that is supplied by a system to a user.
  - Black box—What <something> does.
  - White box—How <something> does it.

## Use Cases: Approach (1)

**Use Case ⇨ Model ⇨ Design**

- **Jacobson *et al.* (1992) – First to apply the concept of use cases to software engineering.**
- **Constantine and Lockwood (1999) – “Use cases have been integrated with success into virtually every approach of OO analysis and design.”**

## Use Cases: Approach (2)

**"...teams that take time and model the problem domain** *by writing use cases* **will plan their programming and ultimately deliver better systems than those that plunge directly into coding."**

Reference: Constantine L. L., 1995. Under Pressure. Software Development, **3** (6).

## Use Cases: Approach (3)

We have written and collected more than 133 use cases.

| *Application* | *U-C Format* | *Count* |
|---|---|---|
| VGL | XML | 33 |
| GenExplorer | HTML tables | 35 |
| Genex | n/a | 0 |
| Folding | n/a | 0 |
| MGX (+ Protex) | MS Word text | 65+ |
| | | ------ |
| | Total | 133+ |

## Use Cases: Example – MGX

**UCID: ASAM.2**
**Name:** Administrator enters VGL.

**Actor:** Administrator.

**Pre-Condition:** Administrator opens a new session or a saved session

**Purpose:** Administrator opens VGL as a stand-alone application.

**Overview:** Administrator indicates that he wants to open VGL as a stand-alone application. MGX starts VGL as a stand-alone application.

**Typical course of events:**
1) Administrator indicates that he wants to open VGL.
2) MGX starts VGL as a stand-alone application.

**Post-Condition:** VGL is running.

## Use Cases: Example – VGL

```
<UseCase>
  <Characteristic>
    <ID>33</ID>
    <Name>Turn off Balloon Help</Name>
    <Actor>Student</Actor>
    <GoalInContext>Turn off Balloon Help</GoalInContext>
    <PreCondition>Balloon Help is on</PreCondition>
    <PostCondition>Balloon Help is off</PostCondition>
    <TriggerEvent>Actor selects Balloon Help</TriggerEvent>
  </Characteristic>
  <Main>
    <Step value="1">
      <Name>Student</Name>
      <Action>selects Balloon Help from Help Menu</Action>
    </Step>
  </Main>
  <Info>
    <Author>Chung Ying Yu</Author>
    <ModifiedBy>David Portman</ModifiedBy>
  </Info>
</UseCase>
```

## Use Cases: Summary

- We have compiled use cases taken from a variety of sources, including the documents written by former CS student teams.
- We have written many new use cases of our own.
- We are discussing how to employ use cases effectively, as part of the software modelling and design process.

## One-Step Build: Definition

A one-step build is a single script that

- Does a full checkout from scratch.
- Compiles every line of code.
- Creates all executables, installation packages, and final media.

## One-Step Build: Why?

A one-step build improves the developers' efficiency by

- Automating the build process.
- Handling all sorts of media in one step.
- Improving consistency and repeatability.
- Saving time and money (especially during the final stages of a project).

## One-Step Build: CRISP

- Complete: recipe list of all ingredients.
- Repeatable: version control.
- Informative: radiates valuable info.
- Schedulable: complete and repeatable.
- Portable: machine-independent.

## One-Step Build: Tools

**Tools for doing a one-step build compile only those modules (of source code) that change.**

- ***make/gnumake.***
- ***nmake* – a make tool developed by Bell Labs, licensed by Lucent®.**
- ***jam* – an open-source software build tool maintained by Perforce Software, Inc.**
- ***ant* – (Ant) a Java-based build tool licensed by the Apache Software Foundation.**

## Ant: Why?

- Ant is suitable for cross-platform applications, such as those written in Java.
- Ant is state-of-the-art – its configuration files are based on XML.
  - Each file holds a project and a target tree for executing tasks.
  - Each file task is run by an object.
  - Each file-task object implements a particular interface and OS.

## Ant: A Simple Build File

- Build files are written in XML.
- Each XML build file contains
  - One project.
  - One [default] target (required).

*Build File*

```
 <project >
      <target>
      </target>
  </project>
```

## Ant: Example

```
<project name="MGX" default="run">
 <target name="compile" description="Compiles the code">
   <javac srcdir="src_1/" destdir="bin/" />
 </target>

 <target name="jarfile" depends="compile"
        description="makes jar file">
   <jar destfile="bin/MGX.jar">
   <manifest>
    <attribute name="Main-Class" value="FoldingWindowGUI" />
   </manifest>
   <fileset dir="bin/" />
   </jar>
   <move file="bin/MGX.jar" todir="." />
 </target>

 <target name="run" depends="jarfile" description="run MGX">
   <java jar="MGX.jar" fork="true" />
 </target>
</project>
```

## Ant: <project>

A <project> has 3 attributes and a set of properties.

| | |
|---|---|
| name | name of the <project>. |
| default | target set of tasks [required]. |
| basedir | directory path. |

```
<project name="MGX" default=run basedir="." >
  <description>
      build file for MGX project
  </description>
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  ...
</project>
```

## Ant: <target>

A </target> has attributes.

| | |
|---|---|
| name | name of this </target>. |
| depends | a list of more </target>s. |
| if | name of property to set. |
| unless | name of property not to set. |
| description | function of this </target>. |

```
<target name="compile" description="Compiles
   the code">
...
</target>
```

## Ant: </target>

**A target can depend on (many) other targets.**

```
<target name="jarfile" depends="compile"
   description="makes jar file">
...
</target>
```

## Ant: Task

- A Task is executable code.
- All Tasks have a common structure.

```
<name attribute1="value1" attribute2="value2"
... />
```

## Ant: Types of Task

- Built-in.
- Optional.

```
<jar destfile="bin/MGX.jar">
  <manifest>
    <attribute name="Main-Class"
        value="FoldingWindowGUI" />
  </manifest>
  <fileset dir="bin/" />
</jar>
<move file="bin/MGX.jar" todir="." />
```

## One-Step Build: Summary

- We will perform a one step build
  - At regular time intervals
  - Including all source codes and documents.
- Ant is a good choice as a tool for performing the one-step build.

## Thank You