The goal of the introduction of generics was to reduce the complexity and size of the EASIK project. Before this we had three separate sections of code that were very similar. They were responsible for overviews, sketches, and views. Views and sketches in particular were almost verbatim so the idea was to replace most of the duplicate code in both with a generic version. Overviews were not involved in this process because there was not as much duplicate code. It may be a good idea to include overviews in generics going forward.

Most classes now use quite a bit of generics which may look ugly at times. For example this is the signature for ModelConstraint ->

```
public abstract class ModelConstraint<F extends ModelFrame<F,GM,M,N,E>, GM extends EasikGraphModel,
 M extends Model<F,GM,M,N,E>, N extends ModelVertex<F,GM,M,N,E>, E extends ModelEdge<F,GM,M,N,E>>
extends ModelVertex<F,GM,M,N,E>
{
```

All of these are required by ModelConstraint since it can work with views and sketches. It is possible that rearranging the code could reduce the size of this, perhaps we could move code around so that constraints wouldn't need to work directly with a graph model, however at the time this is the best solution.

In order to avoid raw type warnings, I have sacrificed having more classes with long lists of generics. I'm undecided as to whether this is a good idea or not. The alternative may make the code more simple, but having that many warnings can drown out valuable information. Here is an example of the types of decisions I was making:

The class AddPullbackConstraintAction could have a similar list of generics to the of ModelConstraint seen above, but it only directly needs one, <F extends ModelFrame>. F is used because we can add PullBackConstraints to ViewFrames or SketchFrames so we need the generic to specify which one we are working with.

```
public class AddPullbackConstraintAction<F extends ModelFrame> extends AbstractAction
{
```

The problem is that AddPullbackConstraint calls the constructor for AddPullbackConstraintState which needs the same generic arguments as ModelConstraint. This leaves me with a choice, I can either add the full list of generics to the AddPullbackConstraint signature or I can accept a warning for a AddPullbackConstraintState raw type.

```
new AddPullbackConstraintState(_theFrame.getMModel(), w)  //gives warning for unparameterized generic
```

This is one example but similar cases are found all over the code. I have chosen to add the full list of generics to all classes that need them, but as mentioned this makes the code look fairly ugly. I have not found a way to refactor the code that would reduce this.

Below is the summary of what the generic code was used for.

ModelStateManager (not abstract)

- Replaced
    - SketchStateManager

ModelFrame (abstract)

- Extended by
    - SketchFrame
    - ViewFrame

Model (abstract)

- Extended by
    - Sketch
    - View

ModelVertex (abstract)

- Replaced
    - ViewVertex
    - SketchVertex

ModelEdge (abstract)

- Extended by
    - View_Edge
    - SketchEdge

ModelPath (not abstract)

- Replaced
    - SketchPath
    - ViewPath

ModelConstraint (not abstract)

- Replaced
    - ViewConstraint
    - Constraint

ModelInfoTreeUI (not abstract)

- Replaced
    - SketchInfoTreeUI
    - ViewInfoTreeUI

easik.model.keys.*

- Replaced easik.sketch.keys
- and is now available to views as well

easik.model.attribute.*
- Replaced easik.sketch.attribute.*
- easik.view.attribute.*

easik.model.util.graph
- Replaced
    - easik.sketch.util.graph
    - easik.view.util.graph
- except for
    - ViewGraphModel
    - SketchGraphModel

Statics do not work with generics, this caused a problem with constraints because most of the checks implemented in constraints used static variables and so were static methods. To avoid these static methods I moved them into Model.java. These are the methods.
- asPullbackConstraint(List<ModelPath<N, E>> paths)
- getProjectionPathFor(ModelPath<N, E> p)
- isSumConstraint(List<ModelPath<N, E>> paths)
- isProductConstraint(List<ModelPath<N, E>> paths)