Customizing Check-Mate – Where Do I Start?

Taylor Anderson
NX Product Manager

Agenda

- Quick Background:
  - Checks and Profiles
  - The Configuration Continuum
- Case Studies and Examples:
  - ICE Checker Templates
    - The all-important **do_check:** attribute
  - How to Approach Coding a Check From Scratch
    - The Thought Process
    - Examples and Questions

In Check-Mate, what is a Check?

- A “check” (aka checker) is a small piece of logic that looks for a particular condition within a model.
- Individual check(er)s may validate anything from layering conventions to drafting standards to various modeling best practices or even techniques for organizing and working with assemblies.

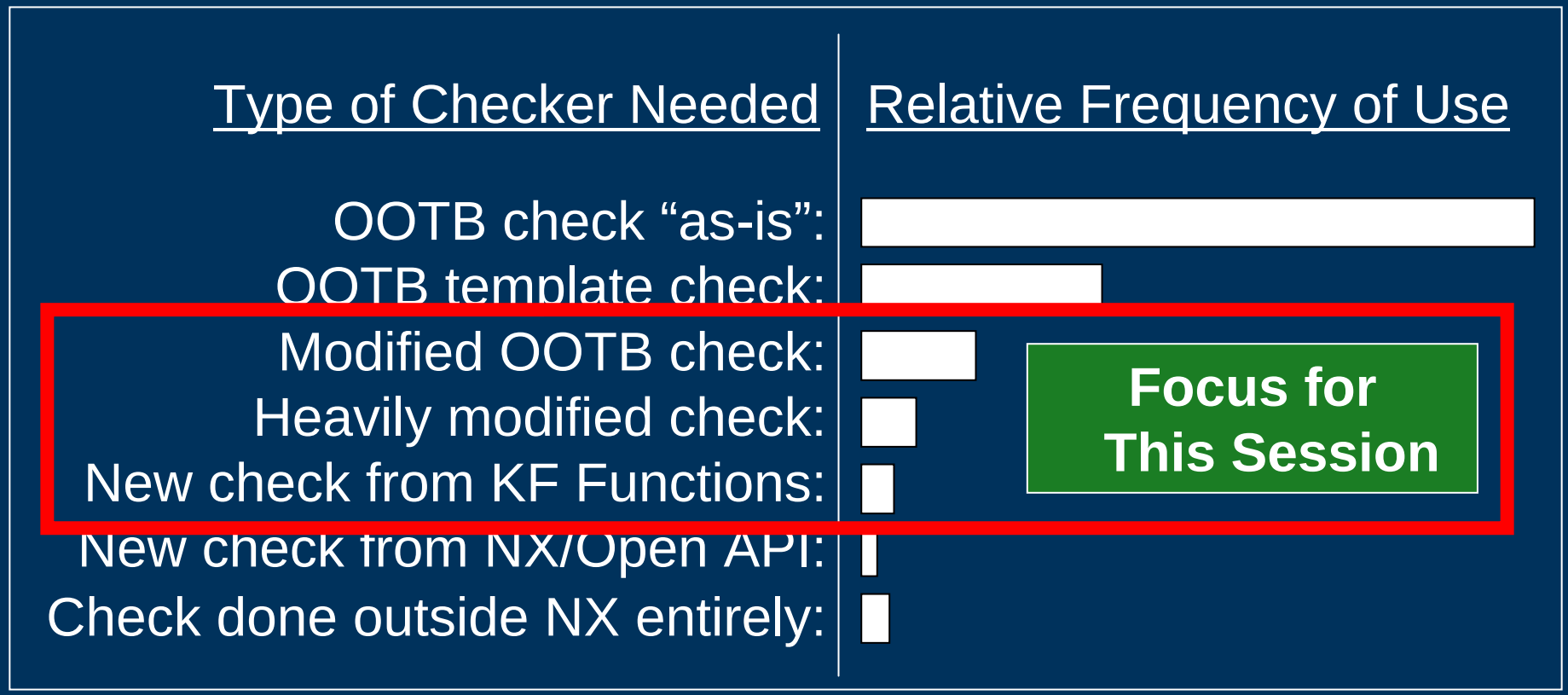
What is a Check-Mate Profile?

- A Check-Mate “profile” is a collection of checks that will be executed together at the same time. Checks contained in a profile can be pre-configured with any default values or needed input parameters.
- A profile is a great tool for ensuring that a complete set of checks is performed using a desired set of quality criteria. It is also a great tool for streamlining the user experience, as it removes the requirement for users to manually collect, configure, and run individual checks.

The Configuration Continuum

Type of Checker Needed	Relative Frequency of Use
OOTB check “as-is”:	
OOTB template check:	
Modified OOTB check:	
Heavily modified check:	
New check from KF Functions:	
New check from NX/Open API:	
Check done outside NX entirely:	

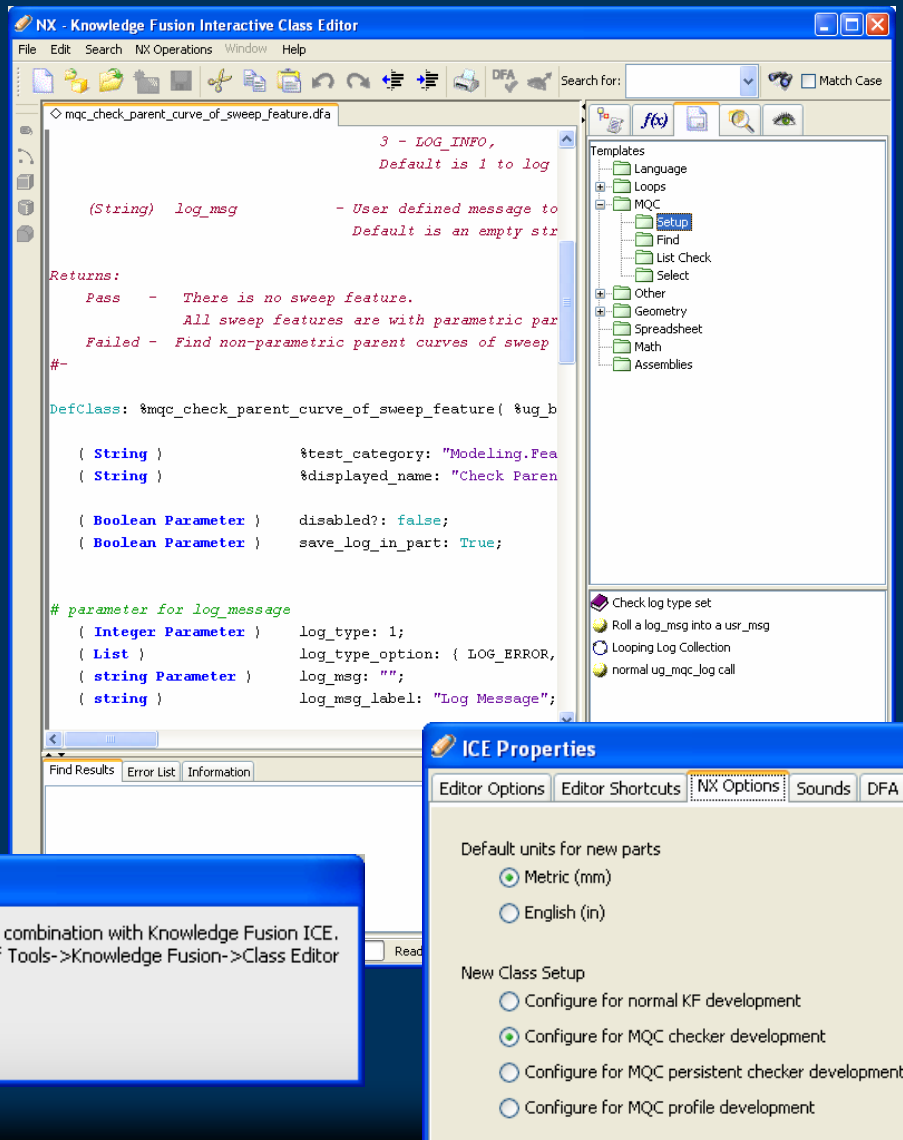
The Configuration Continuum



ICE for Check-Mate

Capability in NX 5

- **Interactive Class Editor** for KF Authoring (new in NX 5)
- ICE has preset “MQC” modes and code templates to simplify the authoring of checks and profiles.



How to tackle one like this…

Desired Check	Required Failure Condition	Reports
Check for layers with entities but without any categories.	Fails when layers have entities but no category.	The layer numbers that have entities but no category.

How to tackle one like this…

Desired Check	Required Failure Condition	Reports
Check for layers with entities but without any categories.	Fails when layers have entities but no category.	The layer numbers that have entities but no category.

Where should I start looking for this?

1. Existing Checkers in the Run Tests UI
2. ICE Search (Classes and Functions)
3. Quick Reference (HTML) in Author Dialog

- What do I want to report?

  A list of layer numbers — `mqc_collect_entity_layers();`

- What subset of layer numbers do I want?

  Layers WITH entities on them AND — `mqc_ask_layer_entities();`

  Layers WITHOUT a category assigned — `mqc_askOrphanLayers();`

Focus in on desired results…

```
(Any Uncached)   do_check:
@{

  $orphan_layers << mqc_askOrphanLayers();


  $entities_in_orphan_layer_list << mqc_ask_layer_entities( $orphan_layers, ALL, False, "lib",
                                                            "libufmqc", "Name",
                                                            "mqc_ufkf_ask_layer_entities" );
```

(List of layers w/o categories)

Focus in on desired results…

```
(Any Uncached)   do_check:
@{

  $orphan_layers << mqc_askOrphanLayers();


  $entities_in_orphan_layer_list << mqc_ask_layer_entities( $orphan_layers, ALL, False, "lib",
                                                            "libufmqc", "Name",
                                                            "mqc_ufkf_ask_layer_entities" );


  $layers_of_orphaned_entities << mqc_collect_entity_layers($entities_in_orphan_layer_list);
```

(List of layers w/o categories)

(List of object tags)

Focus in on desired results…

```
(Any Uncached)   do_check:
@{

  $orphan_layers << mqc_askOrphanLayers();

  $entities_in_orphan_layer_list << mqc_ask_layer_entities( $orphan_layers, ALL, False, "lib",
                                                            "libufmqc", "Name",
                                                            "mqc_ufkf_ask_layer_entities" );

  $layers_of_orphaned_entities << mqc_collect_entity_layers($entities_in_orphan_layer_list);

  $write_log << Loop
  {
    For $layer_num  In  $layers_of_orphaned_entities;
    For $detail_msg Is  "Layer "
                        + Stringvalue( $layer_num )
                        + " is not empty and not in a category.";

    Do ug_mqc_log( LOG_ERROR, {}, $usr_msg + $detail_msg );
  };

};
```

(List of layers w/o categories)

(List of object tags)

(Final list of layers)

NOTE: Calling ug_mqc_log multiple times in one checker is just fine. (Useful for more descriptive messages.)

…or, search more and get lucky. ☺

ug_mqc_askLayerWithoutCategory

Synopsis

```
Defun: ug_mqc_askLayerWithoutCategory( )
 @{CFunc("KF_mqc_ask_layer_without_category","kfmqc");} list
```

Detail:

```
--------------------------------------------------------------------------------
ug_mqc_askLayerWithoutCategory

Description:
   Reports non-empty layers that are not assigned with category

Input:
   None

Return:
      (list)    -  A list of non-empty layers number that are
                   not assigned with category
                   Format:
                   {(integer)without_category_layer1,(integer)without_category_layer2,
                      ...,(integer)without_category_layern }
                   For example:
                   { 5,12,15,28,254 }
--------------------------------------------------------------------------------
```

The right function is simpler…

```
(Any Uncached)   do_check:
@{
  $orphan_layers << mqc_ask_orphan_layers();

  $entities_in_orphan_layer_list << mqc_ask_layer_entities( $orphan_layers, ALL, False, "lib",
                                                            "libufmqc", "Name",
                                                            "mqc_ufkf_ask_layer_entities" );

  $layers_of_orphaned_entities << mqc_collect_entity_layers($entities_in_orphan_layer_list);

  $write_log << Loop
  {
    For $layer_num  In  $layers_of_orphaned_entities;
    For $detail_msg Is  "Layer "
                          + Stringvalue( $layer_num )
                          + " is not empty and not in a category.";

    Do ug_mqc_log( LOG_ERROR, {}, $usr_msg + $detail_msg );
  };

};
```

The right function is simpler…

```
(Any Uncached)   do_check:
@{

  $layers_of_orphaned_entities << ug_mqc_askLayerWithoutCategory( );

  $write_log << Loop
  {
    For $layer_num  In  $layers_of_orphaned_entities;
    For $detail_msg Is  "Layer "
                          + Stringvalue( $layer_num )
                          + " is not empty and not in a category.";

    Do ug_mqc_log( LOG_ERROR, {}, $usr_msg + $detail_msg );
  };

};
```

How about these?

Desired Check	Required Failure Condition	Reports
Check for required entities on specified layers.	Fails when the required entities are missing from the specified layers.	The layer numbers and the missing required entities.
Check for required entities in specified categories.	Fails when the required entities are missing from the specified categories.	The category with layer numbers and the missing required entities.

- What do I want to report?
  - A list of (categories and) layers (and entity descriptions)
- What subset of layer numbers do I want?
  - Layers in the category? `mqc_ask_layers_of_category();`
- What entities do I want to look for?
  - Required ones on the right layers `mqc_collectEntitiesWithFilterOptions();` `mqc_selectEntitiesWithFilters();`
  - Ask by Type or Type AND Name? `mqc_ask_entities_by_type_name();`
  - See also… `mqc_ask_entity_of_category();` `ug_mqc_checkLayerEntityType();`

(UF Types and Subtypes defined in ..\UGCHECKMATE\dfa\mixins\ug_object_types.dfa)

How about these?

Desired Check	Required Failure Condition	Reports
Check for required attribute names.	Fails when any required attribute names are missing	The missing attributes names.

- What do I want to collect?

  The MISSING names.

- What attributes are currently in this part?

  A list of names (of attributes)

  `mqc_ask_part_attributes();`

- Are each of the required ones in the part?

  For each in the required list, is it in the part?

```
Loop {
  For $a In required_names:;
  If ( !member( $a, first(current_names:) ) ) Collect $a;
};
```

if the required name is not in this part…

How about these?

Desired Check	Required Failure Condition	Reports
Check for approved categories	Fails when any unapproved categories are present	The unapproved categories and their corresponding layer numbers.
Check for approved reference sets.	Fails when any unapproved reference sets are present	The unapproved reference sets.
Check for approved attribute names.	Fails when any unapproved attribute names are present.	The unapproved attribute names.

- What do I want to report?

  The EXTRA names in the list. (Reverse of the last one.)

- Are there any extras in the part?

  For each in the part, is it in the approved list?

```
mqc_ask_category();
mqc_ask_all_referencesets();
mqc_ask_part_attributes();
```

```
Loop {
  For $a In first(current_names:);
  If ( !member( $a, approved_names: ) ) Collect $a;
};
```

if this name is not a member of the approved list…

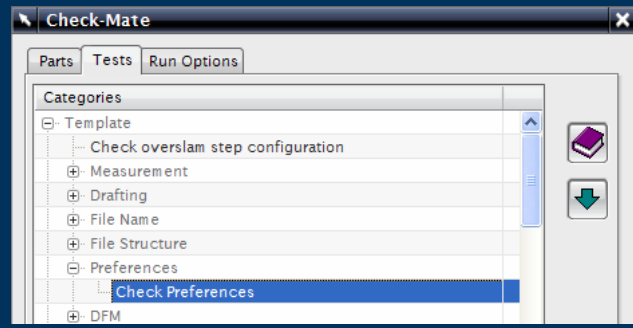
How about these?

Desired Check	Required Failure Condition	Reports
Check for required categories with specified entities.	Fails when the required category is missing from any layer with the specified entity.	The entity and the layer number along with the missing required category.
Check for required secondary categories on layers with a specified primary category.	Fails when the required secondary categories are missing from any layer with the specified primary category.	The primary category and the missing secondary categories with the layer numbers.

- What do I want to report?
  - A list of entities (and some info about them.)
- How would we get there?
  - Start with the specified entities `mqc_askEntities();`
  - Ask their layers `mqc_collect_entity_layers();`
  - See if the layer is in the required category `mqc_askCategoryOfLayer();`
  - (See if the layer is ALSO in the required secondary category) `""`

How about this one?

Desired Check	Required Failure Condition	Reports
"Check Drafting Preferences" Read the company specific default settings and check if any of the settings in the NX file have been changed from the default.	Fail if any Drafting entity settings is not the same as specified in the customer default file.	The Drafting Entities that have been changed from the default.



- What do I want to report?

  A list of entities (and some info about them.)

- What settings do I want to check?

  Wow… A long list. Maybe UGS can help with this one? ☺

```
Checker: %mqc_check_preferences (in mqc_check_preferences.dfa)

Description:

    Verifies that the customer defaults are set properly according
to the defaults specified in the user-defined XML files.
```

Additional Questions:

Q2: I need to check the work layer setting. I found the function "ug_AskWorkLayer" in the KF module but can't find it anywhere in the Check-Mate function list. Does it exist or not? Can we use it? Where is the documentation on it?

A2: Check-Mate is fundamentally based on KF, and so any of the normal KF functions are fair game for use when writing checkers.

Does it exist or not? Yes.
Can we use it? Yes.
Where is the documentation on it? Normal KF docs.

(docs discussion)

Additional Questions:

Q3: I need to check for view dependent entities. My understanding is the function "mqc_isViewDepEntity" checks individual entities. Is there a function that checks the entire model and returns a list of view dependent entitites?

A3: There's a checker in the Drafting category called **`Check View Dependent Geometry`** that will give you back all of the views containing view dependent geometry.

Additional Questions:

Q4: I need to run the "Examine Geometry - Combo" checker on just one solid body, not everything in the entire part file. But I don't want to have to select the solid body interactively but instead specify its entity name, type and layer. Is there an easy way to use the selection capability built into the checker to do this? Or would it be best to integrate the function "**`mqc_selectEntitiesWithFilters`**" to get the entity first?

A4: **`mqc_selectEntitiesWithFilters`** is the way to go for this one. Feed your filtered body to the **`selection:`** attribute, and you should be off and running. Remember that Check-Mate does not store any persistent classes in the part file – there is nowhere to store a selection, for example. Your checker needs to stand alone, and not require any user interaction.

Additional Questions:

Q5: I need to determine which layer a particular entity is on. If I have a list of entity tags that have been obtained through another operation and I want a corresponding list of layers, is the function "mqc_collect_entity_layers" the best one to use?

A5: Yep. That one works great with either a long list or a very short one.

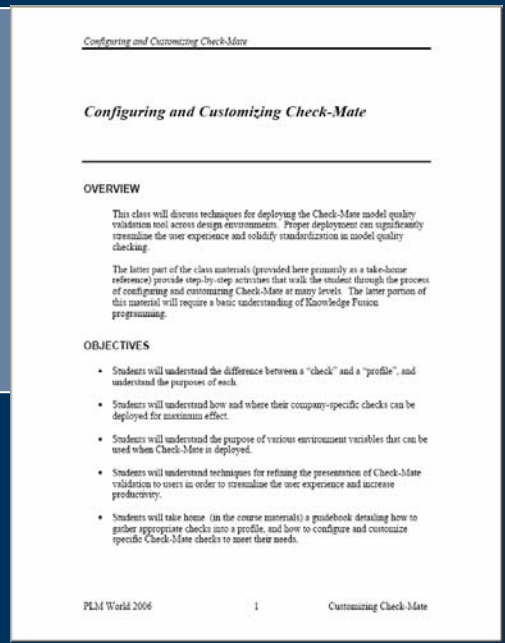
The Configuration Continuum

Type of Checker Needed	Relative Frequency of Use
OOTB check “as-is”:	
OOTB template check:	
Modified OOTB check:	
Heavily modified check:	
New check from KF Functions:	
New check from NX/Open API:	
Check done outside NX entirely:	

Additional Training Material

PLM World 2006 Training Class:

"Configuring and Customizing Check-Mate"



Configuring and Customizing Check-Mate

Configuring and Customizing Check-Mate

OVERVIEW

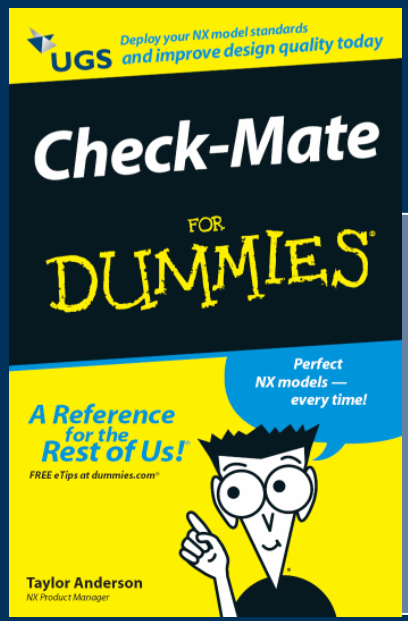
This class will discuss techniques for deploying the Check-Mate model quality validation tool across design environments. Proper deployment can significantly streamline the user experience and solidify standardization in model quality checking.

The latter part of the class materials (provided here primarily as a take-home reference) provide step-by-step activities that walk the student through the process of configuring and customizing Check-Mate at many levels. The latter portion of this material will require a basic understanding of Knowledge Fusion programming.

OBJECTIVES

- Students will understand the difference between a "check" and a "profile", and understand the purposes of each.
- Students will understand how and where their company-specific checks can be deployed for maximum effect.
- Students will understand the purpose of various environment variables that can be used when Check-Mate is deployed.
- Students will understand techniques for refining the presentation of Check-Mate validation to users in order to streamline the user experience and increase productivity.
- Students will take home (in the course materials) a guidebook detailing how to gather appropriate checks into a profile, and how to configure and customize specific Check-Mate checks to meet their needs.

PLM World 2006 1 Customizing Check-Mate



The new "Check-Mate for Dummies" Book

NOTE: Electronic copies are not available due to copyright restrictions.

Ask your Sales Rep for a free copy!

Thank You!

taylor.anderson@ugs.com