KPI Reporting - Application Note

1 Overview

This document describes the workflow for reporting key performance indicators in BioRails using Morphit as the data presentation layer. This document assumes some knowledge of databases and SQL.

Metrics should not be calculated just because they can be calculated. They should be specific and relevant and they should serve a purpose. Good metrics and KPI reports can deliver the following to your organization:

- Process insight
- Aligned goals for the company, its departments and their employees
- Performance and process improvement
- Good quantitative internal and external public relations materials

We can calculate the following sorts of metrics from the data in BioRails:

- Time from request to results
- Time from assay start to completion
- Samples submitted in a period of time
- Samples processed in a period of time
- Time from step X to step Y in process Z - We can model anything as a queue

2 Objective

This document will explain how to extract queue and cascade based metrics from BioRails and present the data graphically in Morphit for easy consumption. After reading this document you should understand the following key concepts:

- Which tables and views store key metric data for queues and cascades
- How to structure a SQL based metrics query based on the data in BioRails
- How to add a SQL based query to BioRails
- How to present metrics data in Morphit

3 Background

3.1 Database Objects

In the near future we plan to expose some metrics reporting capabilities directly through the BioRails interface but this will require some assumptions about which metrics people would like to report e.g. metrics by queue or method, methods at the level of a month , week or year etc.

The current implementation of metrics capture is generic and allows BioRails users to extract metrics at any level/grouping. This means a basic understanding of SQL is required to create queries to extract this information from the database; alternatively you could use the example queries from this document if they meet your needs.

3.1.1 TABLES

There are two key tables in the database that track the history of each queue item and tier item respectively:

- QUEUE_ITEM_HISTORIES

Name	Type	Nullable
ID	NUMBER(38)	
QUEUE_ITEM_ID	NUMBER(38)	Y
PARENT_ID	NUMBER(38)	Y
STATE_ID	NUMBER(38)	Y
PREVIOUS_STATE_ID	NUMBER(38)	Y
NEXT_STATE_ID	NUMBER(38)	Y
DAYS	NUMBER	Y
DAYS_OLD	NUMBER	Y
ENTERED_DAY	NUMBER	Y
ENTERED_AT	DATE	Y
ENTERED_BY_USER_ID	NUMBER(38)	Y
EXITED_DAY	NUMBER	Y
EXITED_AT	DATE	Y
EXITED_BY_USER_ID	NUMBER(38)	Y
EVENT_ACTION_ID	NUMBER(38)	Y
REASON	VARCHAR2(255	CHAR)
COMMENTS	VARCHAR2(1024	CHAR)
CREATED_AT	DATE	Y
UPDATED_AT	DATE	Y

- TIER_ITEM_HISTORIES

Name	Type	Nullable
ID	NUMBER(38)	
EVENT_ACTION_ID	NUMBER(38)	Y
TIER_ITEM_ID	NUMBER(38)	Y
CASCADE_TIER_ID	NUMBER(38)	Y

REQUEST_ITEM_ID	NUMBER(38)	Y
RULE_ACTION_ID	NUMBER(38)	Y
RULE_OUTCOME	VARCHAR2(255)	Y
STATE_ID	NUMBER(38)	Y
PREVIOUS_STATE_ID	NUMBER(38)	Y
PREVIOUS_RULE_ACTION_ID	NUMBER(38)	Y
PREVIOUS_RULE_OUTCOME	VARCHAR2(255)	Y
NEXT_STATE_ID	NUMBER(38)	Y
NEXT_RULE_ACTION_ID	NUMBER(38)	Y
NEXT_RULE_OUTCOME	VARCHAR2(255)	Y
DAYS	NUMBER	Y
DAYS_OLD	NUMBER	Y
ENTERED_DAY	NUMBER	Y
ENTERED_AT	DATE	Y
ENTERED_BY_USER_ID	NUMBER(38)	Y
EXITED_DAY	NUMBER	Y
EXITED_AT	DATE	Y
EXITED_BY_USER_ID	NUMBER(38)	Y
REASON	VARCHAR2(255)	Y
COMMENTS	VARCHAR2(255)	Y
CREATED_AT	DATE	Y
UPDATED_AT	DATE	Y

These tables are normalised and therefore querying them and returning plain-English requires a lot of joins to other tables.

To avoid this complexity we provide some user friendly views that have all those joins built in (see the section below).

3.1.2 VIEWS

There are two views in the database that allow for more user friendly queries of queue and tier items respectively as they are fully joined to all the related tables.

- QUEUE_ITEM_FULL_HISTORIES (available in 5.4)
- TIER_ITEM_FULL_HISTORIES (available in 5.5)

4 SQL Based Metrics Queries

The remainder of this document will focus on queue item metrics but the same principles could be applied to tier item metrics.

4.1 Counting Requests, Queue Items and Experiments

At the most basic level of metrics reporting we may want to track throughput in terms of requests submitted, queue items submitted and experiments completed in a given time period. The query below, which has been implemented as a view, gives an example of how easy it is do this type of query.

```
CREATE OR REPLACE VIEW KPI_QUEUE_COUNTS AS
SELECT QUEUE_ITEM_FULL_HISTORIES.QUEUE_NAME AS Assay,
       Count(DISTINCT QUEUE_ITEM_FULL_HISTORIES.QUEUE_ITEM_ID) AS Num_Samples,
       Count(DISTINCT QUEUE_ITEM_FULL_HISTORIES.REQUEST_ID) AS Num_Requests,
       Count(DISTINCT QUEUE_ITEM_FULL_HISTORIES.TASK_ID) AS Num_Expt
  FROM QUEUE_ITEM_FULL_HISTORIES
 WHERE QUEUE_ITEM_FULL_HISTORIES.ENTERED_AT BETWEEN
       To_Date(‘2016-01-01', 'YYYY-MM-DD') AND
       To_Date(‘2016-12-31', 'YYYY-MM-DD')
   AND QUEUE_ITEM_FULL_HISTORIES.STATE_NAME IN ('published', 'completed')
 GROUP BY QUEUE_ITEM_FULL_HISTORIES.QUEUE_NAME
 ORDER BY QUEUE_ITEM_FULL_HISTORIES.QUEUE_NAME;
```

This query will return an output of queue counts for 2016 like the one shown below:

ASSAY QUEUE	NUM_SAMPLES	NUM_REQUESTS	NUM_EXPT
EDGE Dose Response-001	2655	189	1
EDGE Dose Response-002	1952	209	1
EDGE Dose Response-003	1619	82	1
EDGE Dose Response-004	1109	108	28
EDGE Dose Response-005	969	99	52
EDGE Dose Response-006	750	92	18
EDGE Dose Response-007	647	85	23
EDGE Dose Response-008	865	83	56
EDGE Dose Response-009	871	103	31
EDGE Dose Response-010	568	51	24

The query is simply counting the distinct number of items for each type of entity (queue item, request and experiment) where the date is between two values and then grouping the results by the assay queue name.

4.2 Tracking Queue Items By State

One of the most useful queries we can write is to extract the day (not date) of entry into every state in the queue item state flow. The query below, once again implement as a view, pulls back the day of entry for each of the following states (pending, ordered, active, published) for each queue item for each queue.

```
CREATE OR REPLACE VIEW KPI_QUEUE_ITEM_DETAILS AS
SELECT a.QUEUE_ITEM_ID,
       a.QUEUE_NAME,
       a.REQUESTED_DATE,
       a.REQUESTED_DAY,
       a.ACCEPTED_DAY,
       a.READY_DAY,
       a.COMPLETED_DAY
  FROM (SELECT QUEUE_ITEM_FULL_HISTORIES.QUEUE_ITEM_ID,
               Min(CASE
                     WHEN QUEUE_ITEM_FULL_HISTORIES.STATE_NAME IN ('pending') THEN
                      QUEUE_ITEM_FULL_HISTORIES.QUEUE_NAME
                     ELSE
                      NULL
                   END) queue_name,
               Min(CASE
                     WHEN QUEUE_ITEM_FULL_HISTORIES.STATE_NAME IN ('pending') THEN
                      QUEUE_ITEM_FULL_HISTORIES.ENTERED_AT
                     ELSE
                      NULL
                   END) requested_date,
               Min(CASE
                     WHEN QUEUE_ITEM_FULL_HISTORIES.STATE_NAME IN ('pending') THEN
                      QUEUE_ITEM_FULL_HISTORIES.ENTERED_DAY
                     ELSE
                      NULL
                   END) requested_day,
               Min(CASE
                     WHEN QUEUE_ITEM_FULL_HISTORIES.STATE_NAME IN ('ordered') THEN
                      QUEUE_ITEM_FULL_HISTORIES.ENTERED_DAY
                     ELSE
                      NULL
                   END) accepted_day,
               Min(CASE
                     WHEN QUEUE_ITEM_FULL_HISTORIES.STATE_NAME IN ('active') THEN
                      QUEUE_ITEM_FULL_HISTORIES.ENTERED_DAY
                     ELSE
                      NULL
                   END) ready_day,
               Min(CASE
                     WHEN QUEUE_ITEM_FULL_HISTORIES.STATE_NAME IN
                          ('completed', 'published') THEN
                      QUEUE_ITEM_FULL_HISTORIES.ENTERED_DAY
                     ELSE
                      NULL
                   END) completed_day
          FROM QUEUE_ITEM_FULL_HISTORIES
         GROUP BY QUEUE_ITEM_FULL_HISTORIES.QUEUE_ITEM_ID) a
 WHERE a.ACCEPTED_DAY IS NOT NULL
   AND a.COMPLETED_DAY IS NOT NULL;
```

This query will return an output of days of entry for each state like the one shown below:

QUEUE_ITEM_ID	QUEUE_NAME	REQUESTED_DATE	REQUESTED_DAY	ACCEPTED_DAY	READY_DAY	COMPLETED_DAY
11601	EDGE Dose Response-001	04/03/2015	0	5	7	16
11602	EDGE Dose Response-001	04/03/2015	0	5	7	16
11603	EDGE Dose Response-001	04/03/2015	0	5	7	16
11604	EDGE Dose Response-001	04/03/2015	0	5	7	16
11605	EDGE Dose Response-001	04/03/2015	0	5	7	16
12066	EDGE Dose Response-001	10/03/2015	0	2	2	13
12067	EDGE Dose Response-001	10/03/2015	0	2	2	13
12068	EDGE Dose Response-001	10/03/2015	0	2	2	13
12069	EDGE Dose Response-001	10/03/2015	0	2	2	13
12070	EDGE Dose Response-001	10/03/2015	0	2	2	13
12071	EDGE Dose Response-001	10/03/2015	0	2	2	13
12072	EDGE Dose Response-001	10/03/2015	0	2	2	13
12073	EDGE Dose Response-001	10/03/2015	0	2	2	13
14960	EDGE Dose Response-001	31/03/2015	0	1	1	16

The inner query is pulling out the first entry into a given state for each state for each queue item and then squashing out the null entries using a MIN function. The min also ensures that if a queue item is reset to a previous state we drive the metrics from the first instance not later ones.

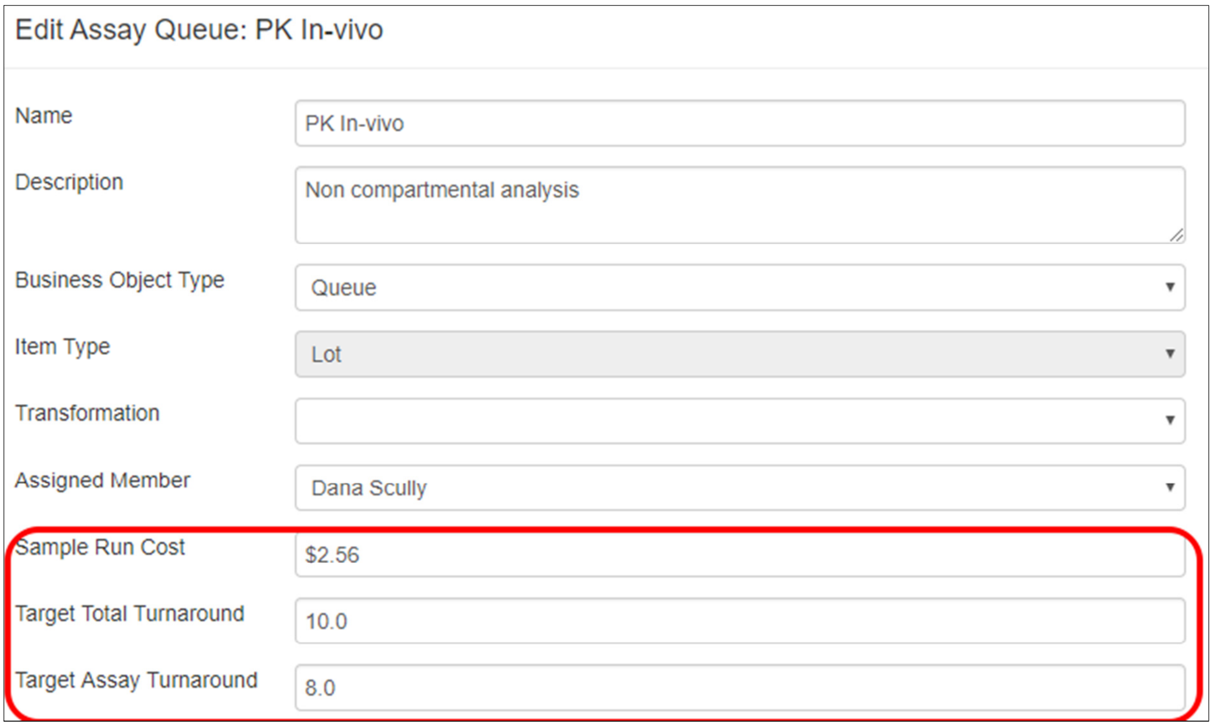
This query is the basis of most of the queries described later in this document. We can do simple calculations to calculate metrics like turnaround time from this set of results e.g. completed_day – requested_day.

4.3 Tracking Turnaround Times

One of the mostly commonly requested metrics by BioRails users is average turnaround times for each queue by month. This can be measured against a statistical factor like standard deviation or median absolute deviation or alternatively against a hard target e.g. target turnaround. BioRails 5 offers the ability to add custom properties to any business object type and this is a convenient way of adding a field to outline queues to track target times.

In the screenshot below we can see 3 custom properties added to a queue:

- Sample Run Cost
- Target Total Turnaround (days)
- Target Assay Turnaround (days)



Edit Assay Queue: PK In-vivo

Name	PK In-vivo
Description	Non compartmental analysis
Business Object Type	Queue
Item Type	Lot
Transformation	
Assigned Member	Dana Scully
Sample Run Cost	$2.56
Target Total Turnaround	10.0
Target Assay Turnaround	8.0

In the section following we will be using a view built from the custom properties values table named PROJECT_ELEMENT_VALUES to use these values in our query.

The view creation SQL is shown below.

```
create or replace view queue_cust_prop_vals as
select a.name as queue_name, b.data_content as target_total_turn, c.data_content as
target_assay_turn, d.data_content as sample_run_cost
  from outline_queues a
  left outer join PROJECT_ELEMENT_VALUE_SUMMARY  b on a.project_element_id =
b.project_element_id and b.name = 'Target Total Turnaround'
  left outer join PROJECT_ELEMENT_VALUE_SUMMARY  c on a.project_element_id =
c.project_element_id and c.name = 'Target Assay Turnaround'
  left outer join PROJECT_ELEMENT_VALUE_SUMMARY  d on a.project_element_id =
d.project_element_id and d.name = 'Sample Run Cost'
```

Using the view above and a query similar to that shown in section 4.2 we can build a query which looks at the average turnaround times by queue and month and compares them to the target times.

The query is multi-layered to allow for multiple group-by statements to be applied and is once again implemented as a view (see below).

```
CREATE OR REPLACE VIEW KPI_TURNAROUND_TIMES AS
SELECT z.queue_name,
       z.year_month,
       z.total_turnaround,
       z.assay_turnaround,
       y.target_total_turn,
       y.target_assay_turn
  FROM (SELECT queue_name,
               to_char(requested_date, 'YYYY-MM') as year_month,
               trunc(Avg(turnarround), 2) as total_turnaround,
               trunc(Avg(assay_turnarround), 2) as assay_turnaround
          FROM (SELECT a.QUEUE_ITEM_ID,
                       a.requested_date,
                       a.queue_name,
                       (a.completed_day - a.requested_day) turnarround,
                       (a.completed_day - a.accepted_day) assay_turnarround
                  FROM (SELECT queue_item_full_histories.QUEUE_ITEM_ID,
                               Min(CASE
                                     WHEN queue_item_full_histories.STATE_NAME IN
                                          ('pending') THEN
                                      queue_item_full_histories.queue_name
                                     ELSE
                                      NULL
                                   END) queue_name,
                               Min(CASE
                                     WHEN queue_item_full_histories.STATE_NAME IN
                                          ('pending') THEN
                                      queue_item_full_histories.ENTERED_AT
                                     ELSE
                                      NULL
                                   END) requested_date,
                               Min(CASE
                                     WHEN queue_item_full_histories.STATE_NAME IN
                                          ('pending') THEN
                                      queue_item_full_histories.ENTERED_DAY
                                     ELSE
                                      NULL
                                   END) requested_day,
                               Min(CASE
                                     WHEN queue_item_full_histories.STATE_NAME IN
                                          ('ordered') THEN
                                      queue_item_full_histories.ENTERED_DAY
                                     ELSE
                                      NULL
                                   END) accepted_day,
                               Min(CASE
                                     WHEN queue_item_full_histories.STATE_NAME IN
                                          ('active') THEN
                                      queue_item_full_histories.ENTERED_DAY
                                     ELSE
                                      NULL
                                   END) ready_day,
                               Min(CASE
                                     WHEN queue_item_full_histories.STATE_NAME IN
                                          ('completed', 'published') THEN
                                      queue_item_full_histories.ENTERED_DAY
                                     ELSE
                                      NULL
                                   END) completed_day
                          FROM queue_item_full_histories
                         GROUP BY queue_item_full_histories.QUEUE_ITEM_ID) a
                 WHERE a.accepted_day IS NOT NULL
                   AND a.completed_day IS NOT NULL)
         GROUP BY queue_name, to_char(requested_date, 'YYYY-MM')
         ORDER BY queue_name, to_char(requested_date, 'YYYY-MM')) z
  LEFT OUTER JOIN QUEUE_CUST_PROP_VALS y
    ON z.queue_name = y.queue_name;
```

The query outputs data like that shown in the table below:

QUEUE_NAME	YEAR_MONTH	TOTAL_TURNAROUND	ASSAY_TURNAROUND	TARGET_TOTAL_TURN	TARGET_ASSAY_TURN
EDGE Dose Response-001	2015-03	15.74	11.21	18	12
EDGE Dose Response-001	2015-04	30.3	18.59	18	12
EDGE Dose Response-001	2015-05	23.36	21.71	18	12
EDGE Dose Response-001	2015-06	13.42	10.91	18	12
EDGE Dose Response-001	2015-07	13.32	7.66	18	12
EDGE Dose Response-001	2015-08	22.83	18.44	18	12
EDGE Dose Response-001	2015-09	19.55	13.16	18	12
EDGE Dose Response-001	2015-10	11.89	11.75	18	12
EDGE Dose Response-001	2015-11	28.75	24.25	18	12
EDGE Dose Response-001	2015-12	61.65	57	18	12
EDGE Dose Response-001	2016-01	109.9	105.1	18	12
EDGE Dose Response-001	2016-02	14.56	8.41	18	12
EDGE Dose Response-001	2016-03	11.88	10.14	18	12
EDGE Dose Response-001	2016-04	16.54	16.11	18	12
EDGE Dose Response-001	2016-05	25.03	13.8	18	12
EDGE Dose Response-002	2015-03	20.5	19.42	26	20
EDGE Dose Response-002	2015-04	15.31	11.8	26	20

4.4 Tracking Full Queue Statistics

We can extend the approach in the previous query to extract full statistics from a queue based on total and assay turnaround times by month and queue. These are all the statistics needed to create a box plot per month for each queue.

```
CREATE OR REPLACE VIEW KPI_QUEUE_STATISTICS AS
SELECT z.queue_name,
       EXTRACT(YEAR FROM TO_DATE(z.year_month,'YYYY-MM'))as "Year",
       EXTRACT(MONTH FROM TO_DATE(z.year_month,'YYYY-MM'))as "Month",
       min(z.turnarround) as min_total_turnaround,
       min(z.assay_turnarround) as min_assay_turnaround,
       max(z.turnarround) as max_total_turnaround,
       max(z.assay_turnarround) as max_assay_turnaround,
       avg(z.turnarround) as avg_total_turnaround,
       avg(z.assay_turnarround) as avg_assay_turnaround,
       median(z.turnarround) as med_total_turnaround,
       median(z.assay_turnarround) as med_assay_turnaround,
       stddev(z.turnarround) as stdev_total_turnaround,
       stddev(z.assay_turnarround) as stdev_assay_turnaround,
       min(z.TURN_BOX_TOP) as turn_box_top,
       min(z.TURN_BOX_BOTTOM) as turn_box_bottom,
       min(z.ASSAY_BOX_TOP) as assay_box_top,
       min(z.ASSAY_BOX_BOTTOM) as assay_box_bottom
  FROM (SELECT a.QUEUE_ITEM_ID,
               TO_CHAR(a.requested_date, 'YYYY-MM') as year_month,
               a.queue_name,
               (a.completed_day - a.requested_day) turnarround,
               (a.completed_day - a.accepted_day) assay_turnarround,
               PERCENTILE_DISC(0.75) WITHIN GROUP(ORDER BY (a.completed_day -
a.requested_day) asc) OVER(PARTITION BY a.queue_name, TO_CHAR(a.requested_date, 'YYYY-
MM')) "TURN_BOX_TOP" ,
               PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY(a.completed_day -
a.requested_day) asc) OVER(PARTITION BY a.queue_name, TO_CHAR(a.requested_date, 'YYYY-
MM')) "TURN_BOX_BOTTOM",
               PERCENTILE_DISC(0.75) WITHIN GROUP(ORDER BY (a.completed_day -
a.accepted_day) asc) OVER(PARTITION BY a.queue_name, TO_CHAR(a.requested_date, 'YYYY-MM'))
"ASSAY_BOX_TOP" ,
               PERCENTILE_DISC(0.5) WITHIN GROUP(ORDER BY(a.completed_day -
a.accepted_day) asc) OVER(PARTITION BY a.queue_name, TO_CHAR(a.requested_date, 'YYYY-MM'))
"ASSAY_BOX_BOTTOM"
          FROM (SELECT queue_item_full_histories.QUEUE_ITEM_ID,
                       Min(CASE
                             WHEN queue_item_full_histories.STATE_NAME IN
                                  ('pending') THEN
                              queue_item_full_histories.queue_name
                             ELSE
                              NULL
                           END) queue_name,
                       Min(CASE
                             WHEN queue_item_full_histories.STATE_NAME IN
                                  ('pending') THEN
                              queue_item_full_histories.ENTERED_AT
                             ELSE
                              NULL
                           END) requested_date,
                       Min(CASE
                             WHEN queue_item_full_histories.STATE_NAME IN
                                  ('pending') THEN
                              queue_item_full_histories.ENTERED_DAY
                             ELSE
                              NULL
                           END) requested_day,
                       Min(CASE
                             WHEN queue_item_full_histories.STATE_NAME IN
                                  ('ordered') THEN
                              queue_item_full_histories.ENTERED_DAY
                             ELSE
                              NULL
                           END) accepted_day,
                       Min(CASE
                             WHEN queue_item_full_histories.STATE_NAME IN
                                  ('active') THEN
                              queue_item_full_histories.ENTERED_DAY
                             ELSE
                              NULL
                           END) ready_day,
                       Min(CASE
                             WHEN queue_item_full_histories.STATE_NAME IN
                                  ('completed', 'published') THEN
                              queue_item_full_histories.ENTERED_DAY
                             ELSE
                              NULL
                           END) completed_day
                  FROM queue_item_full_histories
                 GROUP BY queue_item_full_histories.QUEUE_ITEM_ID) a
         WHERE a.accepted_day IS NOT NULL
           AND a.completed_day IS NOT NULL) z
 GROUP BY z.queue_name, z.year_month
 ORDER BY z.queue_name, z.year_month;
```

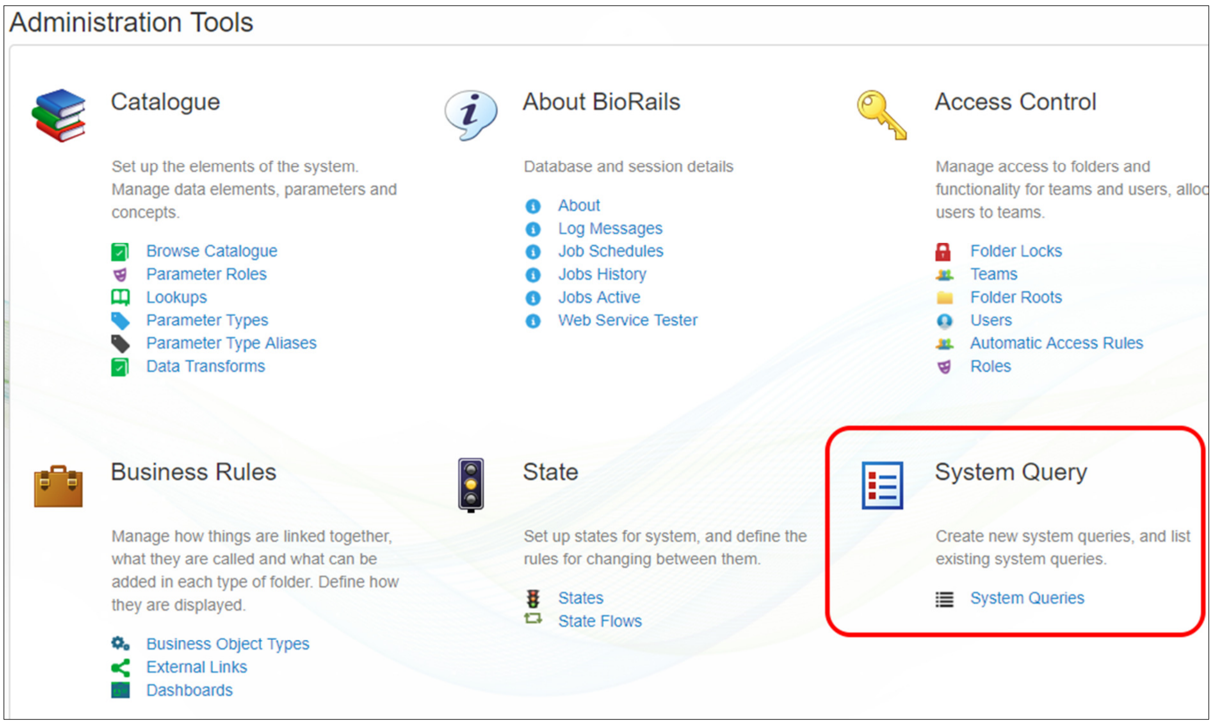
The query outputs data like that shown in the table below. The query outputs a significant number of columns therefore the results of one row of the query have been pivoted to allow them to be displayed on the page.

QUEUE_NAME	EDGE Dose Response-001
Year	2015
Month	3
MIN_TOTAL_TURNAROUND	2
MIN_ASSAY_TURNAROUND	2
MAX_TOTAL_TURNAROUND	17
MAX_ASSAY_TURNAROUND	15
AVG_TOTAL_TURNAROUND	15.74
AVG_ASSAY_TURNAROUND	11.21
MED_TOTAL_TURNAROUND	17
MED_ASSAY_TURNAROUND	11
STDEV_TOTAL_TURNAROUND	3.51
STDEV_ASSAY_TURNAROUND	2.75
TURN_BOX_TOP	17
TURN_BOX_BOTTOM	17
ASSAY_BOX_TOP	11
ASSAY_BOX_BOTTOM	11

5 BioRails Queries

5.1 Adding a SQL Query to BioRails

System queries, SQL or otherwise, are added to BioRails from the administration tools dialog (see below).



Administration Tools

Catalogue

Set up the elements of the system. Manage data elements, parameters and concepts.

- Browse Catalogue
- Parameter Roles
- Lookups
- Parameter Types
- Parameter Type Aliases
- Data Transforms

About BioRails

Database and session details

- About
- Log Messages
- Job Schedules
- Jobs History
- Jobs Active
- Web Service Tester

Access Control

Manage access to folders and functionality for teams and users, alloc users to teams.

- Folder Locks
- Teams
- Folder Roots
- Users
- Automatic Access Rules
- Roles

Business Rules

Manage how things are linked together, what they are called and what can be added in each type of folder. Define how they are displayed.

- Business Object Types
- External Links
- Dashboards

State

Set up states for system, and define the rules for changing between them.

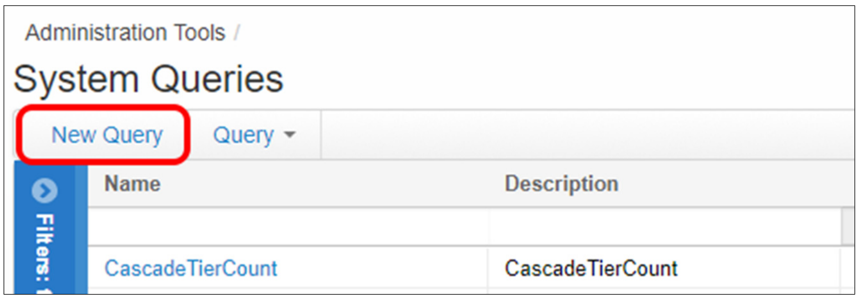
- States
- State Flows

System Query

Create new system queries, and list existing system queries.

- System Queries

Clicking the system queries link will take you to a report of all the queries defined in the system. The new query button (shown below) will launch a dialog which allows you to define a system query.

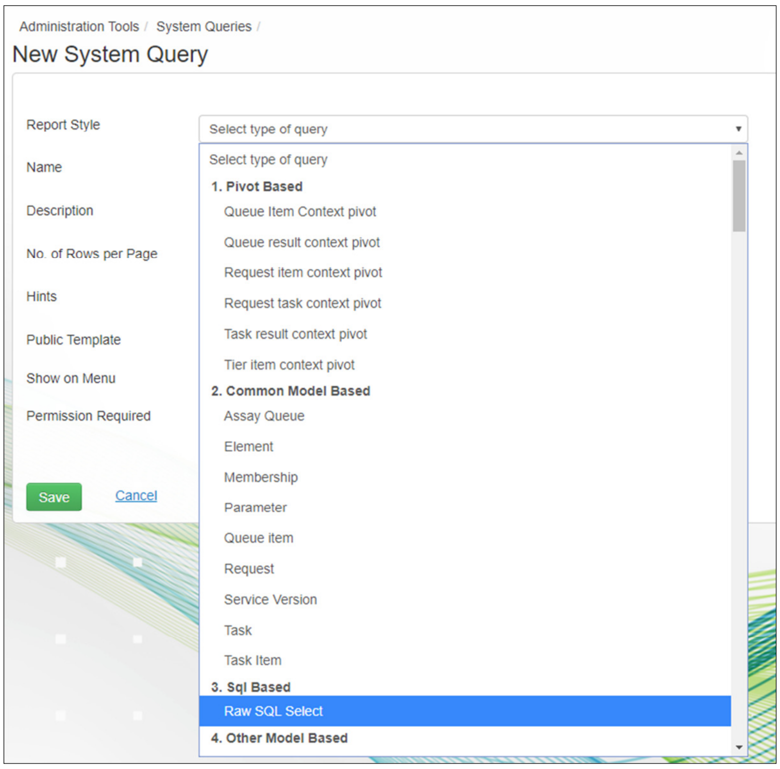


Administration Tools /

System Queries

Name	Description
CascadeTierCount	CascadeTierCount

In order to define a SQL query we need to select "Raw SQL Select" from the "Report Style" drop down (see below).



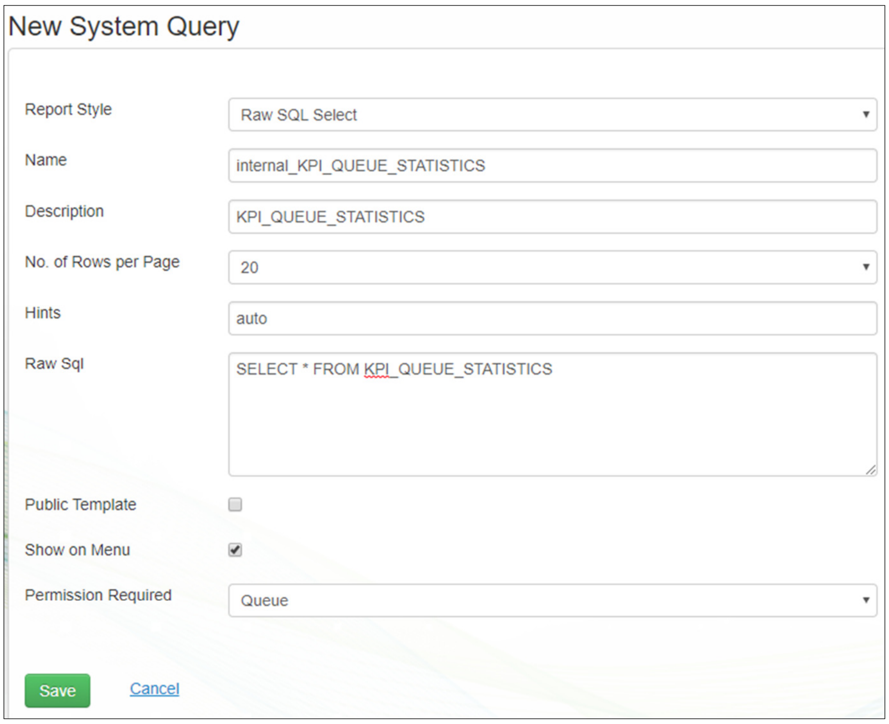
Administration Tools / System Queries /

New System Query

Report Style	Select type of query
Name	
Description	
No. of Rows per Page	
Hints	
Public Template	
Show on Menu	
Permission Required	

Select type of query
1. Pivot Based
   - Queue Item Context pivot
   - Queue result context pivot
   - Request item context pivot
   - Request task context pivot
   - Task result context pivot
   - Tier item context pivot
2. Common Model Based
   - Assay Queue
   - Element
   - Membership
   - Parameter
   - Queue item
   - Request
   - Service Version
   - Task
   - Task Item
3. Sql Based
   - Raw SQL Select
4. Other Model Based

After selecting "Raw SQL Select" the dialog will add a box for defining the SQL statement which will be labelled "Raw SQL".



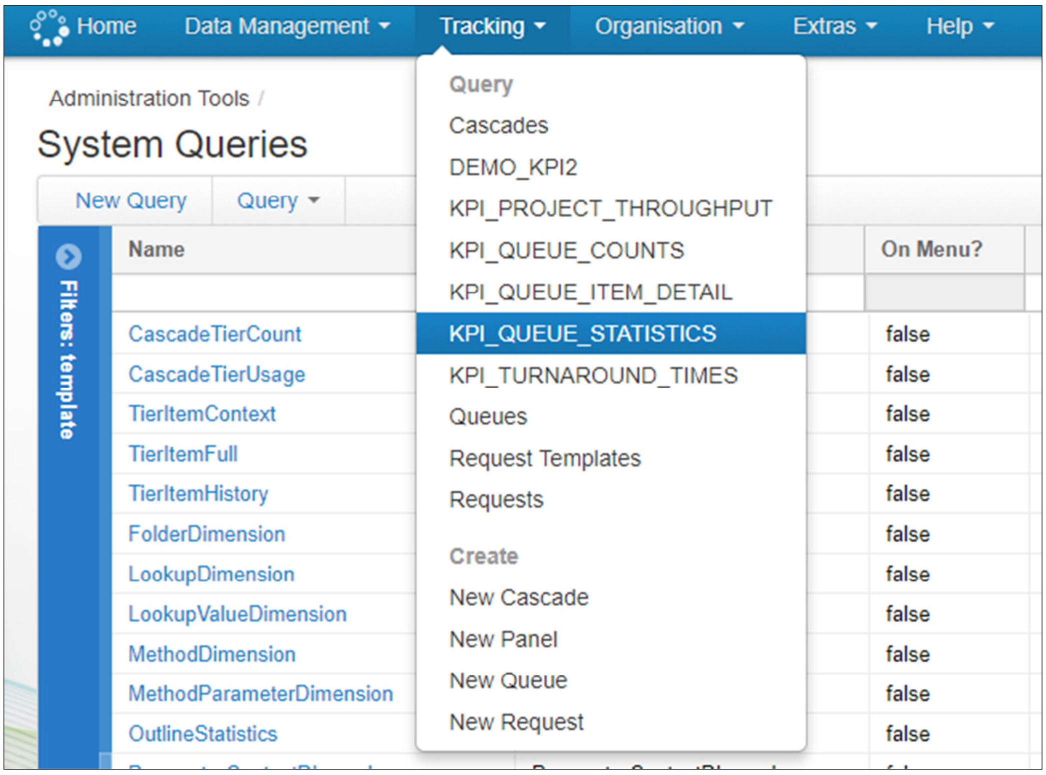
New System Query

Report Style	Raw SQL Select
Name	internal_KPI_QUEUE_STATISTICS
Description	KPI_QUEUE_STATISTICS
No. of Rows per Page	20
Hints	auto
Raw Sql	SELECT * FROM KPI_QUEUE_STATISTICS
Public Template	☐
Show on Menu	✓
Permission Required	Queue

Once the SQL has been added to the form e.g "SELECT * FROM KPI_QUEUE_STATISTICS" (note we should not include a semi-colon at the end of the statement) we can complete the rest of the form.

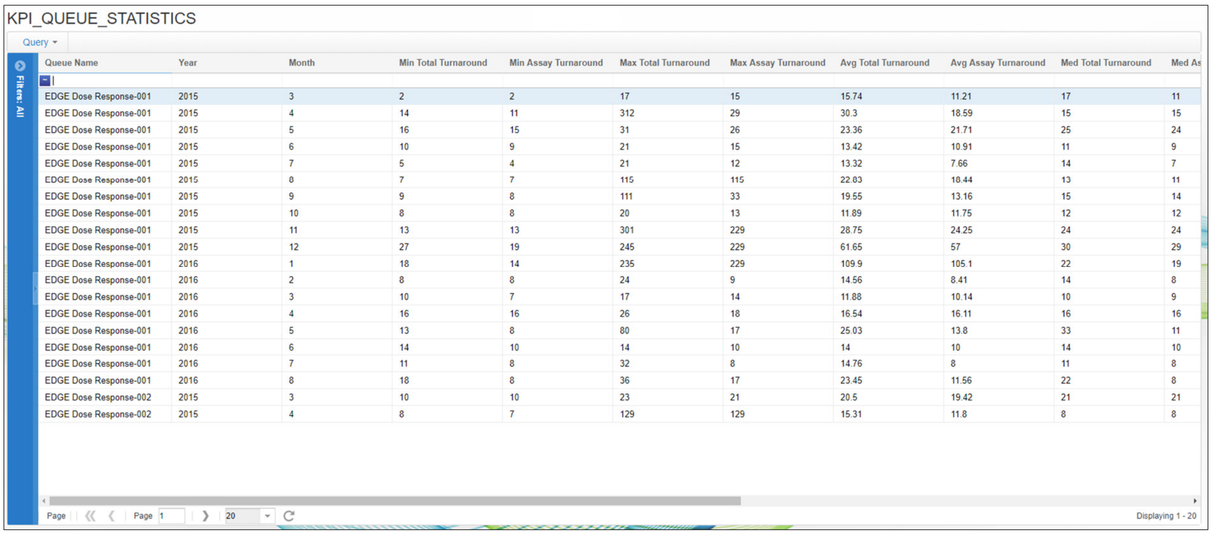
The other fields are described below:

- Name - the internal name of the query. This will not appear on the system menu.
- Description - the external name of the query. This will appear on the system menu.
- No. of Rows Per Page - the default number of rows to appear on each page of results.
- Hints - any SQL hints for the Oracle optimizer.
- Public Template – controls if other users can use this query to make their own.
- Show on Menu – controls if the query will appear on the system menu (see below)
- Permission Required – controls what permissions are needed to run the query and also which entry on the system menu the query will appear on. (In this case tracking).



Name	On Menu?
CascadeTierCount	false
CascadeTierUsage	false
TierItemContext	false
TierItemFull	false
TierItemHistory	false
FolderDimension	false
LookupDimension	false
LookupValueDimension	false
MethodDimension	false
MethodParameterDimension	false
OutlineStatistics	false

The query can then be run by clicking on the menu entry and will display in a standard BioRails query results view.



KPI_QUEUE_STATISTICS

Queue Name	Year	Month	Min Total Turnaround	Min Assay Turnaround	Max Total Turnaround	Max Assay Turnaround	Avg Total Turnaround	Avg Assay Turnaround	Med Total Turnaround	Med As
EDGE Dose Response-001	2015	3	2	2	17	15	15.74	11.21	17	11
EDGE Dose Response-001	2015	4	14	11	312	29	30.3	18.59	15	15
EDGE Dose Response-001	2015	5	16	15	31	26	23.36	21.71	25	24
EDGE Dose Response-001	2015	6	10	9	21	15	13.42	10.91	11	9
EDGE Dose Response-001	2015	7	5	4	21	12	13.32	7.66	14	7
EDGE Dose Response-001	2015	8	7	7	115	115	22.83	18.44	13	11
EDGE Dose Response-001	2015	9	9	8	111	33	19.55	13.16	15	14
EDGE Dose Response-001	2015	10	8	8	20	13	11.89	11.75	12	12
EDGE Dose Response-001	2015	11	13	13	301	229	28.75	24.25	24	24
EDGE Dose Response-001	2015	12	27	19	245	229	61.65	57	30	29
EDGE Dose Response-001	2016	1	18	14	235	229	109.9	105.1	22	19
EDGE Dose Response-001	2016	2	8	8	24	9	14.56	8.41	14	8
EDGE Dose Response-001	2016	3	10	7	17	14	11.88	10.14	10	9
EDGE Dose Response-001	2016	4	16	16	26	18	16.54	16.11	16	16
EDGE Dose Response-001	2016	5	13	8	80	17	25.03	13.8	33	11
EDGE Dose Response-001	2016	6	14	10	14	10	14	10	14	10
EDGE Dose Response-001	2016	7	11	8	32	8	14.76	8	11	8
EDGE Dose Response-001	2016	8	18	8	36	17	23.45	11.56	22	8
EDGE Dose Response-002	2015	3	10	10	23	21	20.5	19.42	21	21
EDGE Dose Response-002	2015	4	8	7	129	129	15.31	11.8	8	8

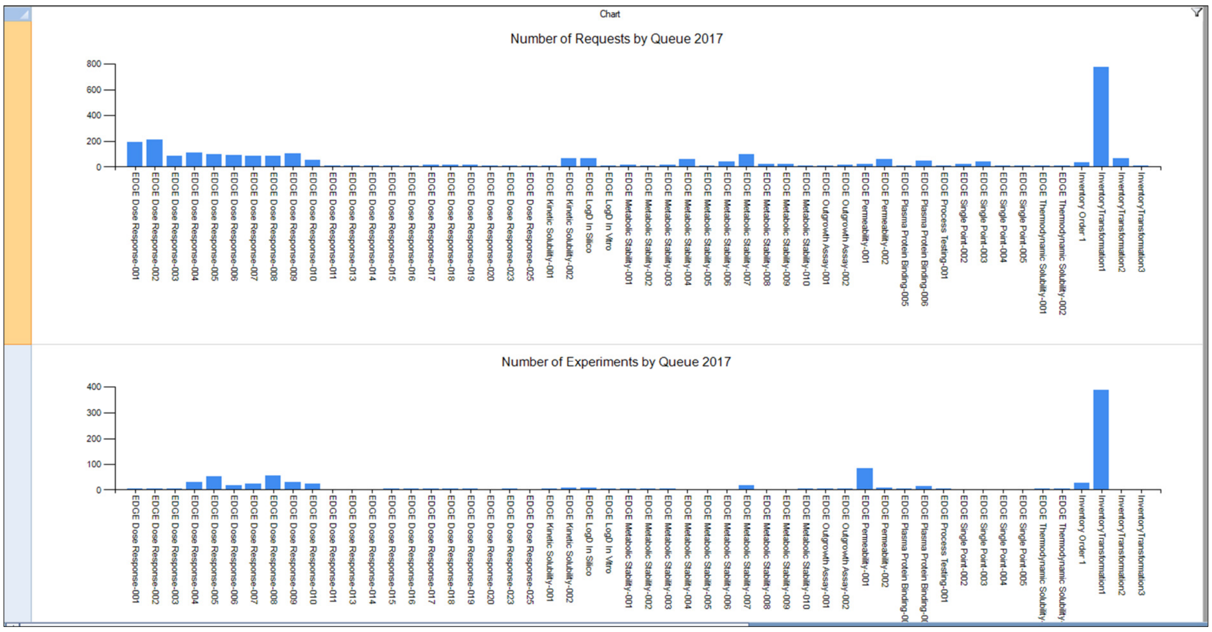
Once the query has been added it can also be accessed through Morphit without the need for an Oracle client or any other reporting tool e.g. Spotfire, D360 etc. etc.

6 Morphit KPI Dashboards

This document is not intended as a Morphit training course and as such the following section will show the types of dashboards which can be created without going through the detail of how the tables will created based on the BioRails queries or how the charts were configured.

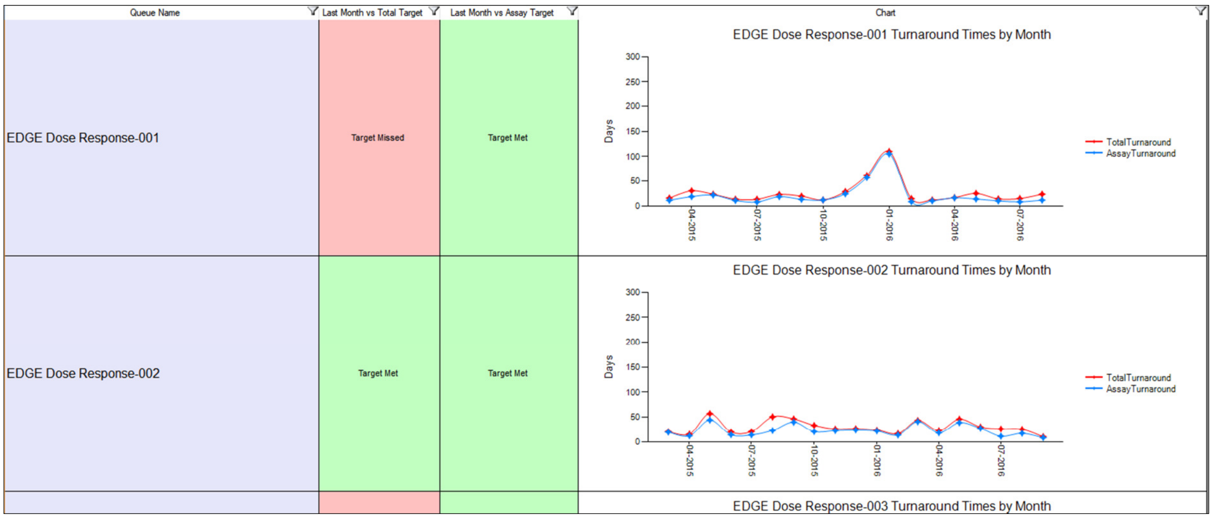
6.1 Throughput Dashboards in Morphit

Simple histograms can be used to display the throughput for all the assay queues across the organization for a given period of time. We could also filter down the time period if desired.



6.2 Simple Trend Dashboards in Morphit

A simple spline or line graph can be used to plot turnaround times by month. Adding traffic light formatting to cells in the report based on the last month's performance against a target or relative statistical measure can simplify review of the data (see below).



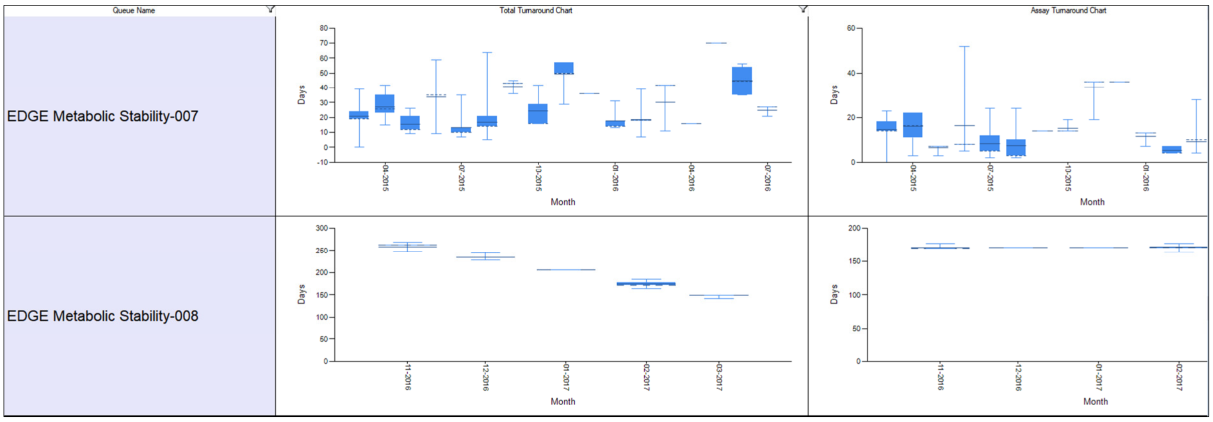
Queue Name	Last Month vs Total Target	Last Month vs Assay Target	Chart
EDGE Dose Response-001	🟥 Target Missed	🟩 Target Met	EDGE Dose Response-001 Turnaround Times by Month
EDGE Dose Response-002	🟩 Target Met	🟩 Target Met	EDGE Dose Response-002 Turnaround Times by Month
	🟥	🟩	EDGE Dose Response-003 Turnaround Times by Month

6.3 Detailed Statistics Dashboards in Morphit

Using a box and whisker plot allows for review of all of the following statistics in a single chart:

- Minimum Turnaround Time
- Maximum Turnaround Time
- Mean Turnaround Time
- Median Turnaround Time
- Lower Quartile Turnaround Time
- Upper Quartile Turnaround Time

The dashboard below shows a plot of these statistics by month and queue for both assay and total turnaround times in days.



Queue Name	Total Turnaround Chart	Assay Turnaround Chart
EDGE Metabolic Stability-007		
EDGE Metabolic Stability-008		