STREAMLINING THE TARGET FABRICATION REQUEST AT THE NATIONAL IGNITION FACILITY (NIF)

C. Manin, E. Bond, A. Casey, R. Clark, G. Norman
Lawrence Livermore National Laboratory, Livermore, CA 94550 USA

Abstract
The NIF Shot Data Systems (SDS) team developed the Target Request Tool (TRT) Web application for facilitating the management of target requests from creation to approval. TRT provides a simple-to-use and user-friendly interface that allows the user to create, edit, submit and withdraw requests. The underlying design uses the latest Web technologies such as Node.js, Express, jQuery and JavaScript. The overall software architecture and functionality will be presented in this paper.

INTRODUCTION

National Ignition Facility (NIF) targets are complex engineering marvels in tiny packages. Creating them requires interplay among target designers, materials scientists, and precision engineers. The laser drives a target capsule inward at nearly a million miles an hour. Because the targets are subjected to extreme temperatures (greater than those in the Sun) and pressures (similar to those found in the core of Jupiter) during experiments, the targets must be designed, fabricated, and assembled with extreme precision [1].

The target production lifecycle begins with submission of a formal target request. Experimentalists and project engineers create the target feature definition based on dozens of existing and/or new parameters determined by the physics requirements and the type of shot. When the shot calls for an existing target, a previous target fabrication request can be duplicated. However, when it calls for a new type of target, a new request must be created. And in those cases, supporting documentation must be provided describing the custom parts that will be needed in the target build.

Before the commissioning of this project, experimentalists, project engineers and target fabrication team members ("users") first utilized a tool developed in Apex (Oracle Application Express). This application was developed as part of an existing tool suite called Production Optics Reporting and Tracking (PORT). The PORT-based target request tool had three major limitations: underlying data architecture precluded future automation in target order processing, data was usually duplicated, and page loading times were very slow.

Given the above limitations of the PORT-based tool, and with an estimated 500 targets needed to be produced each year, it became clear that users urgently needed a new tool. The decision was made to develop a completely new application versus modifying the existing one.

APPLICATION REQUIREMENTS

A brief description of the core requirements for the TRT is provided below:

1. Allow the requester to link the target request with an experimental planning ID called Facility and Laser Integrated Planning (FLIP) ID. This feature is critical since every target that is built must be destined for use on an experiment.
2. Display shot planning data. These data include metadata about the experimental team, diagnostic instrumentation, and target fielding parameters.
3. Allow the requester to select target features from a menu. Feature selection can either be started from a blank template or with a pre-populated set of features that the requester chooses.
4. Allow the requester to enter custom target features. Allow the user to select 'Other' from the options and enter a comment to describe the customized feature.
5. Provide the ability to upload attachments.
6. Provide target request status. Inform the user on the status of the request and drive when specific panels of the application can be edited.
7. Implement roles and permissions. Enable or disable UI capabilities (e.g., changing a field, modifying target request (TR) status) given the user role.
8. Provide target orders search capability (with filters). Provide a list of all orders and ability to filter by certain parameters, e.g., TR number, status, requester, etc.

CHOICE OF TECHNOLOGY

The needs from the Target Fabrication organization resulted in a schedule that allowed for only four months of development time. This limited development time was a key factor when selecting the technologies for this project. We decided to work with modern Web technologies that were familiar to the team and that would allow for reuse of software from other SDS tools.

Node.js

We chose Node.js for the back-end because it is fast to implement, is modern technology, and is supported by a large community of developers. It also pairs well with Web technologies we currently use and allows us to seamlessly connect to existing databases. Node.js is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side [2].

It has the following characteristics:
- Uses V8 engine by Google.
- Is a good fit for real-time applications.
- Is suitable for non-CPU-intensive operations.
- Provides an effective single codebase with JavaScript in server and client, making it easy to send and synchronize data between these two points.

• Provides access to an array of open-source code through its package manager (npm).

Express

This technology was an obvious choice because it is considered the *de facto* framework for Node.js applications. It is a minimal and flexible Node.js web application framework [3] written in JavaScript. Also, it is free and open-source software.

It has the following characteristics:

• Helps manage everything, from routes to handling requests, views and errors.

• Provides a thin layer of fundamental Web application features without obscuring Node.js features.

• Has a myriad of HTTP utility methods and middleware that have access to the request and response objects.

• Has multiple methods for querying the request and constructing the result.

JavaScript and Kendo UI

As JavaScript was selected for the back-end, JavaScript was also chosen for the front-end. To complement JavaScript, we selected Kendo UI for the user interface. Kendo UI is a commercial off-the-shelf product that provides more than 70 customizable UI components. We chose this technology because it shortened development time, provided a nice look-and-feel to the user interface, and was already being used by other SDS applications. Some of the Kendo UI components used were grids (with filters, sorting, page count, and search), buttons, dropdown menus, and date text fields with calendar selector, among others.

Docker

This technology provided rapid deployment to all environments (development, QA, and production) and image portability (the same image could be deployed to all environments). It also gave us the option to roll back a release if needed without having to run a new code build. In addition, we could have multiple containers from different applications running on the same machine, allowing us to have multiple tools running in the same server machine.

ARCHITECTURE

The TRT architecture is composed of three main pieces (see Fig. 1).

Back-end

The back-end was built using Node.js and Express. We used the Node.js-required file "package.json" for listing application dependencies and scripts for start, build, and clean.

The basic routing was done by creating an instance of Express (*var app = express()*) and using the following structure: *app.http-request-method(path, handler)*. Individual pages were given their own path, i.e.: '/TRT/ view-all-orders' for displaying all TR orders.

The Web server behavior and URL configurations were done with the help of a few Node.js and Express middleware modules: 'body-parser' for handling JSON, Raw, Text and URL encoded form data, and 'express.static' for serving static files such as images and third-party libraries.

The templating language we used for the view engine was Embedded JavaScript (EJS). It is very easy to use, complies with the Express view system, and allows us to have nested views. At runtime, the template engine replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client [4].

The handling of data was done with routes/web services by creating a router object (*var router = express.Router()*) and adding middleware and HTTP method routes in the form of: *router.get(path, [callback, ...] callback)* for pulling data, and *router.post(path, [callback, ...] callback)* for inserting data. If the response is successful, the resulting data are sent to the front-end as a JSON object. Otherwise, errors are handled and logged.

Database

Data managed by TRT are stored in an Oracle database. The schema used for the PORT-based application had to be refactored to avoid data duplication and take advantage of the current target fabrication process. Communication of the back-end with the database was done with "node-oracledb" and "orawrap." The former creates the connection to the database and the latter creates a listening pool on the provided port. When querying data, orawrap methods take an SQL command and parameter value(s) (if any) to generate the results. This is done using the *execute()* method embedded in the body of the router methods (shown above).

Front-end

The front-end was developed with a model-view-controller pattern using JavaScript, jQuery, Kendo UI, Bootstrap, HTML, and CSS. This pattern was selected to provide a clear separation between view and logic, to easily subdivide the UI into multiple sections and panels, and to provide flexibility to divide the work among developers. This section separation was made according to the type of data displayed. The result is an interactive UI with panels that appear from left to right building the sections gradually after the user's selection.

The view is mainly composed by a mix of HTML items ("div", "table" and "list"). The model is created from a set of mapping files for each of the UI sections where the HTML elements in the view are mapped to fields in the database (HTML ID to database column). The controller handles field updates, registers event handlers, loads and injects templates, and renders panels.

The most basic view (the first panel on the left is displayed) happens when a new TR is created. From there, there is a minimum amount of menu options the user must select to save the TR. This initial state is the "Draft" state, which means that the TR is still in progress and the user can close the TR and open it later to continue working before submitting it. Also, only the creator and a member of the experimental team have the required permissions to edit the TR. All permissions are handled by the UI using the role information obtained from the back-end and logic applied to each panel.

The full UI view is composed of the following sections: Shot Pairing, Shot Planner Data, Target Menu, and Target Status (see Fig. 2). The resulting application is a website that runs in Chrome and Firefox.

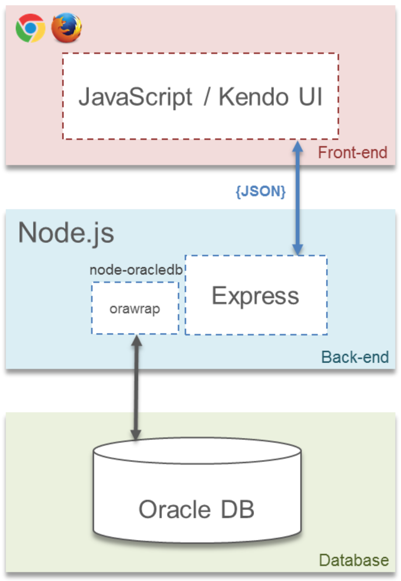


Figure 1: Architecture overview.

BUILD AND DEPLOYMENT

We used Docker for deploying TRT to the Development, QA and Production environments. To facilitate this, we used Git and Atlassian tools (Bitbucket, Bamboo and Jira) for full build and deployment integration.

Bamboo allowed for predefining the build and deployment plans, so that once a build succeeds, it is ready to create a Docker image for deployment to a temporary machine. In this machine, the Docker image is saved to a tar file with a single-line command and retrievable from all three environments. From a particular environment, a script is run to create and start the Docker container derived from the previous image. This gives the flexibility to deploy a container to one machine at a time and run different releases in all three if needed. Docker also provides the benefits of being able to easily revert to older versions, stop/delete containers, and share the same server machines with other applications.

CURRENT STATUS AND FUTURE WORK

TRT is currently used by more than 50 users on a regular basis. Older TRs have been ported and are accessible through TRT. It is hosted in the internal NIF site together with other applications. Launching the application is done by clicking the “Target Orders (TRT)” link under the “Targets & Orders” menu. It is actively maintained and supported by the SDS team. Since its first release, a few additions have been made, such as automatic generation of TRs when a new experiment is created and automatic logging of user actions that affect the state of the TR.

Some work will be needed to modify the current back-end code that uses orawrap. At the time of writing, the orawrap library is no longer being maintained. It has been added to the core Oracle database driver (node-oracledb).

CONCLUSION

We have developed a software tool that supports a more streamlined target fabrication process. The tool provides faster loading time, great user interaction, and data integration. The use of modern technologies allowed the software team to meet the overall project goals primarily within the development time allocated.

ACKNOWLEDGEMENT

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-CONF-739192.

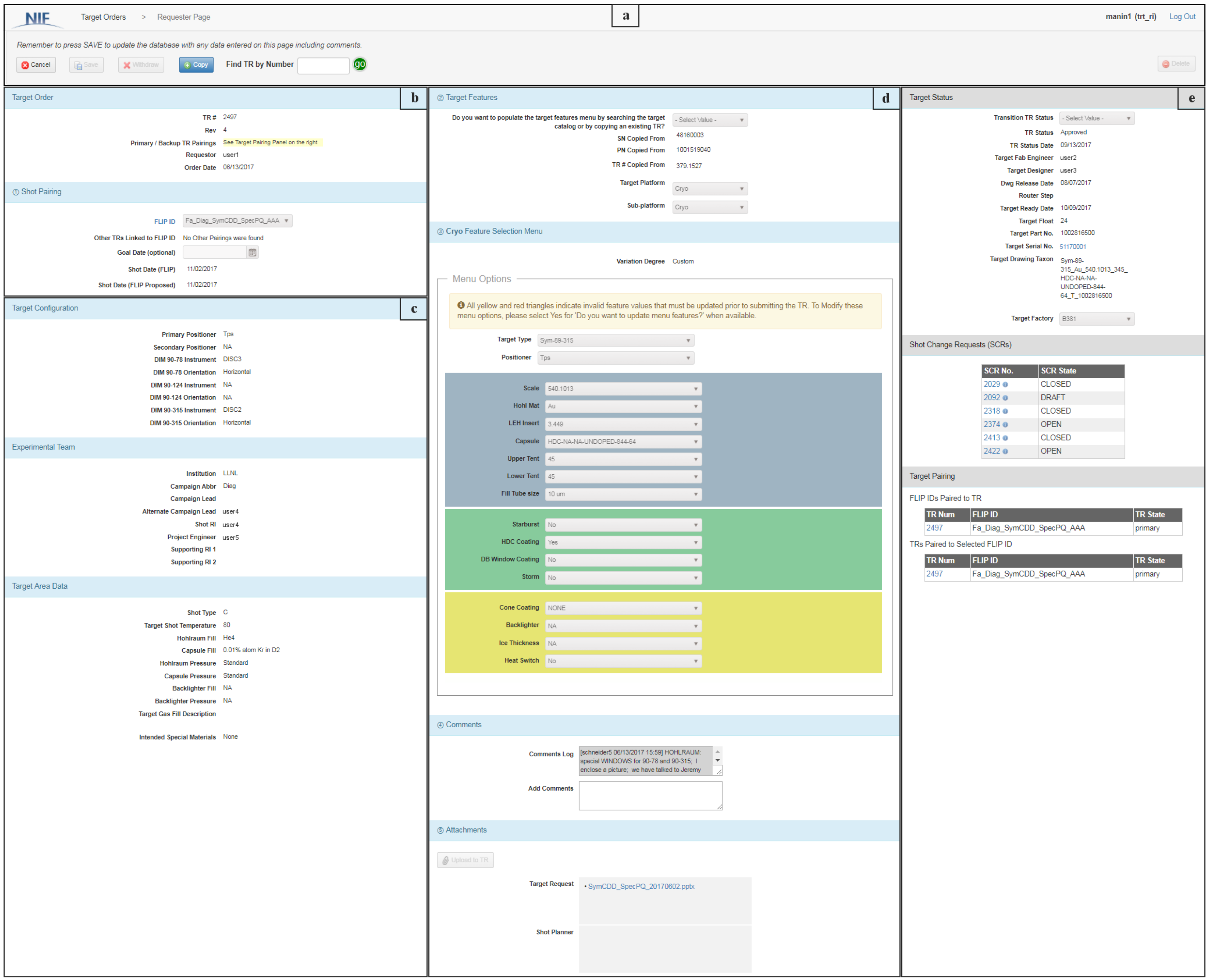
REFERENCES

[1] NIF Target Fabrication, https://lasers.llnl.gov/about/how-nif-works/seven-wonders/target-fabrication

[2] Node.js Wiki, https://en.wikipedia.org/wiki/Node.js

[3] Express Wiki, https://en.wikipedia.org/wiki/Express.js

[4] Express template guide, http://expressjs.com/en/guide/using-template-engines.html



Remember to press SAVE to update the database with any data entered on this page including comments.

Target Order

TR #	2497
Rev	4
Primary / Backup TR Pairings	See Target Pairing Panel on the right
Requestor	user1
Order Date	06/13/2017

Shot Pairing

FLIP ID	Fa_Diag_SymCDD_SpecPQ_AAA
Other TRs Linked to FLIP ID	No Other Pairings were found
Goal Date (optional)	
Shot Date (FLIP)	11/02/2017
Shot Date (FLIP Proposed)	11/02/2017

Target Configuration

Primary Positioner	Tps
Secondary Positioner	NA
DIM 90-78 Instrument	DISC3
DIM 90-78 Orientation	Horizontal
DIM 90-124 Instrument	NA
DIM 90-124 Orientation	NA
DIM 90-315 Instrument	DISC2
DIM 90-315 Orientation	Horizontal

Experimental Team

Institution	LLNL
Campaign Abbr	Diag
Campaign Lead	
Alternate Campaign Lead	user4
Shot RI	user4
Project Engineer	user5
Supporting RI 1	
Supporting RI 2	

Target Area Data

Shot Type	C
Target Shot Temperature	80
Hohlraum Fill	He4
Capsule Fill	0.01% atom Kr in D2
Hohlraum Pressure	Standard
Capsule Pressure	Standard
Backlighter Fill	NA
Backlighter Pressure	NA
Target Gas Fill Description	
Intended Special Materials	None

Target Features

Do you want to populate the target features menu by searching the target catalog or by copying an existing TR?	- Select Value -
SN Copied From	48160003
PN Copied From	1001519040
TR # Copied From	379.1527
Target Platform	Cryo
Sub-platform	Cryo

Cryo Feature Selection Menu

Variation Degree Custom

Menu Options

All yellow and red triangles indicate invalid feature values that must be updated prior to submitting the TR. To Modify these menu options, please select Yes for 'Do you want to update menu features?' when available.

Target Type	Sym-89-315
Positioner	Tps
Scale	540.1013
Hohl Mat	Au
LEH Insert	3.449
Capsule	HDC-NA-NA-UNDOPED-844-64
Upper Tent	45
Lower Tent	45
Fill Tube size	10 um
Starburst	No
HDC Coating	Yes
DB Window Coating	No
Storm	No
Cone Coating	NONE
Backlighter	NA
Ice Thickness	NA
Heat Switch	No

Comments

Comments Log: [schneider5 06/13/2017 15:59] HOHLRAUM: special WINDOWS for 90-78 and 90-315; I enclose a picture; we have talked to Jeremy

Add Comments

Attachments

Upload to TR

Target Request • SymCDD_SpecPQ_20170602.pptx

Shot Planner

Target Status

Transition TR Status	- Select Value -
TR Status	Approved
TR Status Date	09/13/2017
Target Fab Engineer	user2
Target Designer	user3
Dwg Release Date	08/07/2017
Router Step	
Target Ready Date	10/09/2017
Target Float	24
Target Part No.	1002816500
Target Serial No.	51170001
Target Drawing Taxon	Sym-89-315_Au_540.1013_345_HDC-NA-NA-UNDOPED-844-64_T_1002816500
Target Factory	B381

Shot Change Requests (SCRs)

SCR No.	SCR State
2029	CLOSED
2092	DRAFT
2318	CLOSED
2374	OPEN
2413	CLOSED
2422	OPEN

Target Pairing

FLIP IDs Paired to TR

TR Num	FLIP ID	TR State
2497	Fa_Diag_SymCDD_SpecPQ_AAA	primary

TRs Paired to Selected FLIP ID

TR Num	FLIP ID	TR State
2497	Fa_Diag_SymCDD_SpecPQ_AAA	primary

Figure 2: An approved TR is shown. (a)Top menu with navigation links, user information and log-out button, TR search field, and action buttons. The main sections are: (b)Shot Pairing, (c)Shot Planner Data, (d)Target Menu, and (e)Target Status.