Local, OCLC, RAPID?

Integrating Fulfillment with Server Add-ons

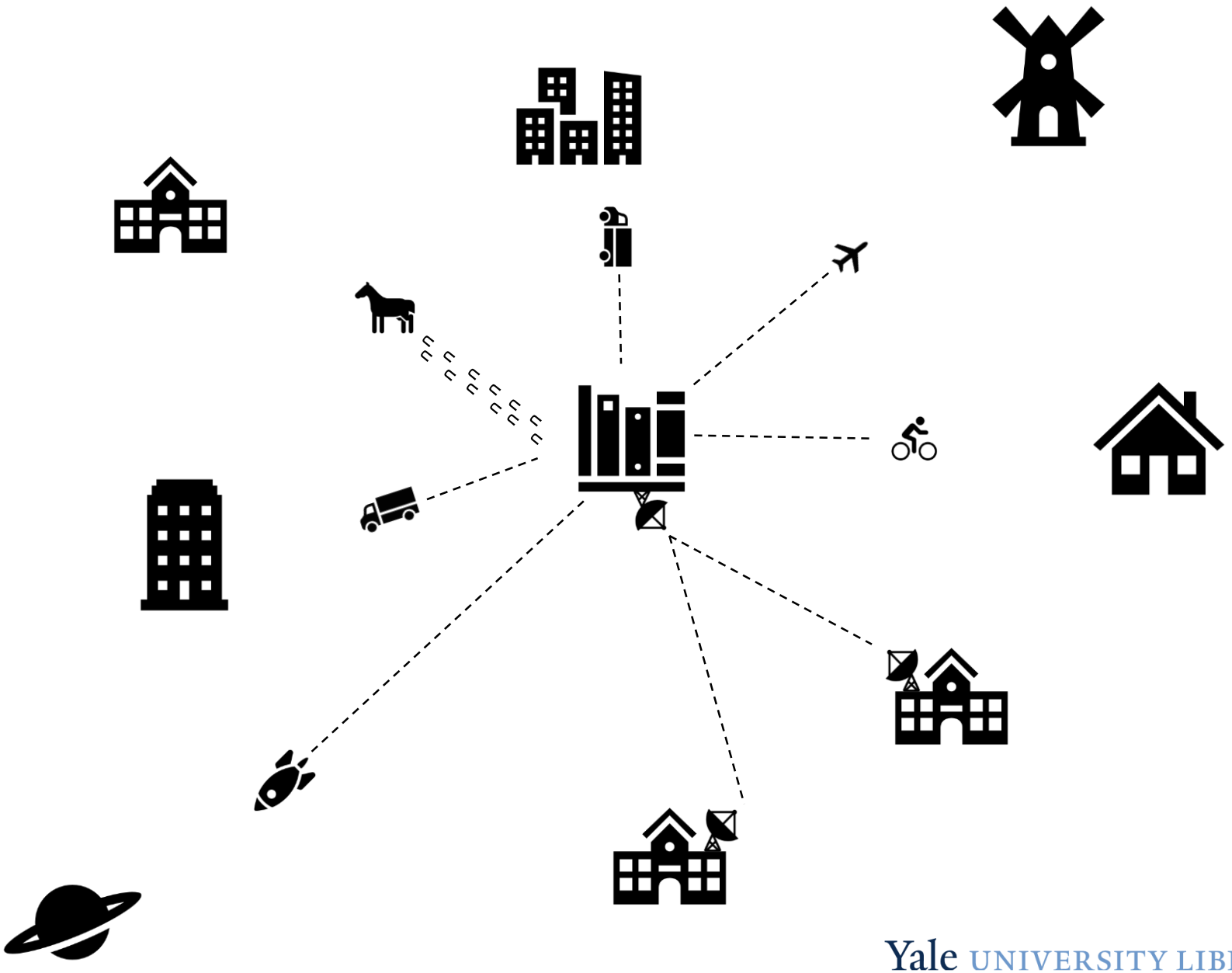
16 March 2017

Steelsen Smith

Yale University Library

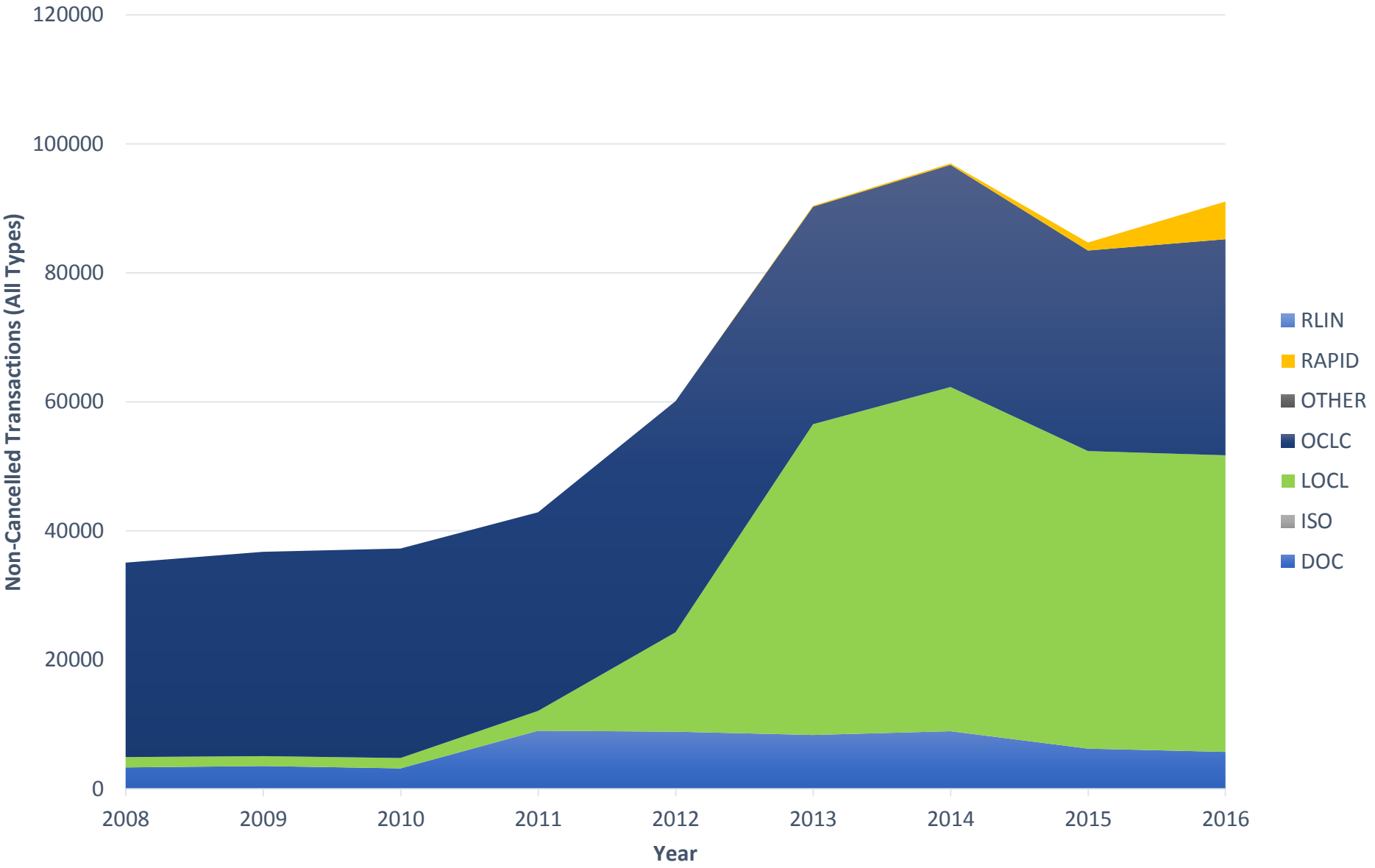
Future of resource sharing?

More!



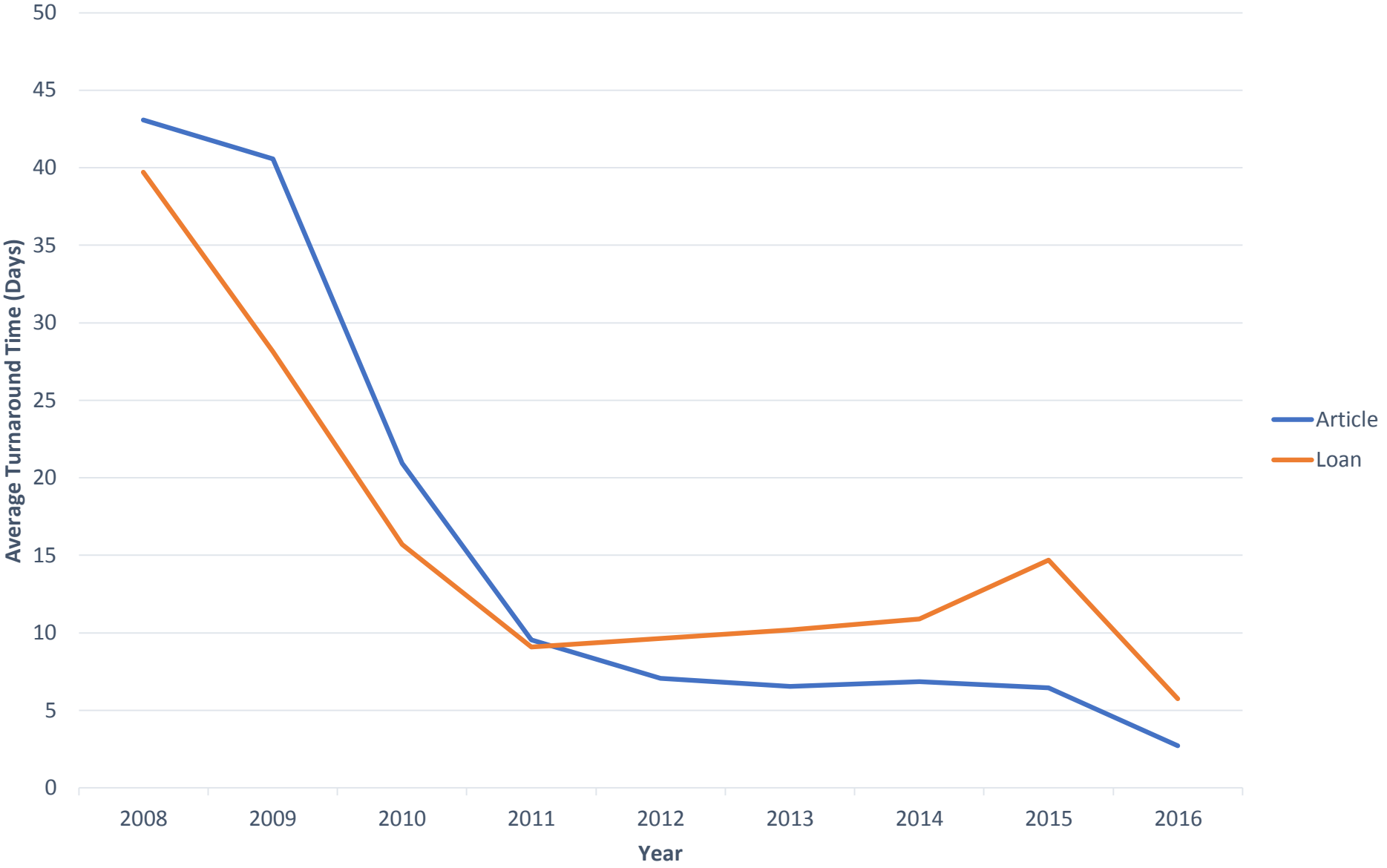
“Special” relationships and services – too successful?

Resource Sharing Volume by System



Yale UNIVERSITY LIBRARY

Average Turnaround Time, All Transactions, All Outcomes



Rethinking Access Services

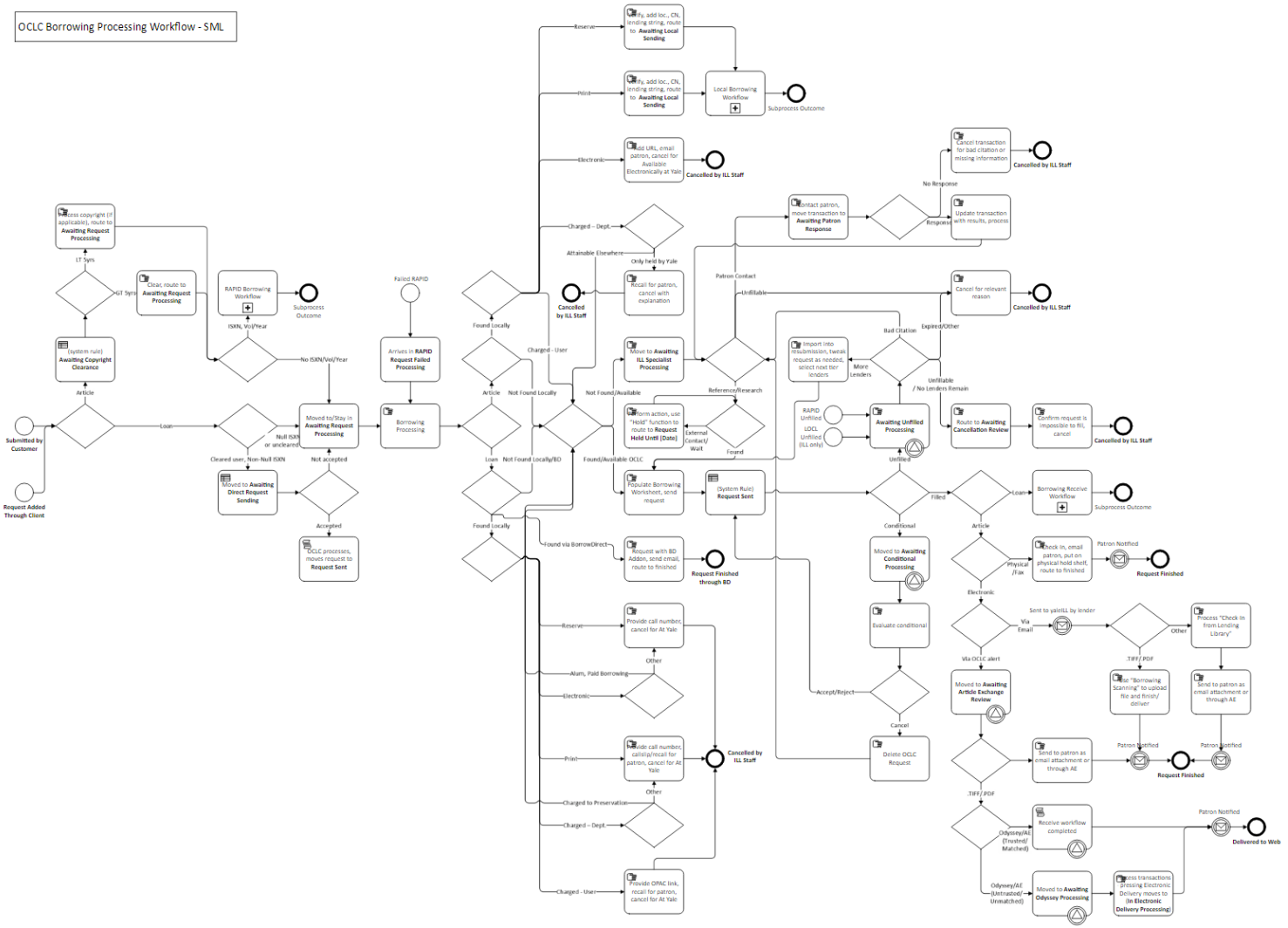
- Focus on workflows
  - Process first, destination second – don't worry about system unless necessary
- Consolidate services
  - Creatively use what you've built
- Be involved
  - Dashboards don't replace managers
- Automate judiciously
  - Prioritize by impact – try to find things that machines do well


ORANGE BLEND
FOUR Loko

Types of Automation

- External application
  - Shift a workflow outside of ILLiad
  - No constraints since you're not using ILLiad
  - **BUT** Challenges at the points of integration
- Script or Custom Application
  - Use an independent program to modify ILLiad
  - "Super user" – effect changes directly upon transactions without limits
  - **BUT** Custom development that is tightly coupled to specific ILLiad versions
- Server Add-on
  - Use ILLiad's extension framework
  - Ready-made tools to interact with transactions
  - **BUT** constrained by framework limits

Choose Path Based on Workflow



Choose Path Based on Workflow

- Use an external application when
  - You need on-demand access to specifically formatted information
  - You need to control all aspects of interaction
  - You need a custom GUI
- Use scripts and custom applications when
  - You need to modify protected fields in ILLiad in a way that may damage stability if done by a server addon
  - You need to take action on a filesystem which ILLiad cannot access
- Use Server Add-ons
  - For everything else
- Remember – there are also client addons – transaction, user, and main form!

What We Did

- Consolidated print (physical) lending, including consortial agreements, to centralize lending shipping and processing
- Consolidate scanning for all services, including Scan and Deliver, OCLC Articles, RAPID, and Course Reserves – all are treated as lending articles and follow identical workflows within central library and storage facility
- Maintain scanning operations in branch libraries
- Implemented automated routing, retries, and cancellation for Scan and Deliver requests

Challenges Faced

- Increased volume
- Requests that may not correspond to appropriate fulfillment system
  - ILL request form for an object held locally
  - Local scan request for an item we do not own
  - Request for an item available through separate consortium
- Mingled processing output
  - Consolidated scanning improves efficiency but blends all transactions together
  - Consolidated retrieval does the same
  - On-site scanning is more efficient but universal ILLiad training is impractical
- Complex consortial arrangements
  - Member of BorrowDirect – consortium using Relais for all activity
  - Certain branches members of free reciprocal groups (e.g., ATLA)
  - Policies vary between branches even for local transactions

Using Automation to Integrate Disparate Services

- Identify transactions requiring intervention early
  - Cancel automatically and identify those transactions needing special attention quickly
- Offsite Scanning Integration
  - Allow offsite storage (library shelving facility) to scan for all transaction types (including OCLC, RAPID, and Local) and deliver directly without training staff in ILLiad.
- File cleanup
  - Pre-screen outgoing scans for potential issues and modify delivery mechanism to reduce errors (e.g., automatically switch to RAPID's own delivery mechanism - RAPIDx)
- Manage special billing arrangements for branch libraries
  - Handle free reciprocal borrowing that only applies to specific branch libraries
- Move transactions into local fulfillment from RAPID and OCLC
  - Discover ILL requests for items held locally and route them into Scan and Deliver (document delivery)
- Move transactions out of local fulfillment into RAPID and OCLC
  - Automate handling of items requested locally but not on shelf

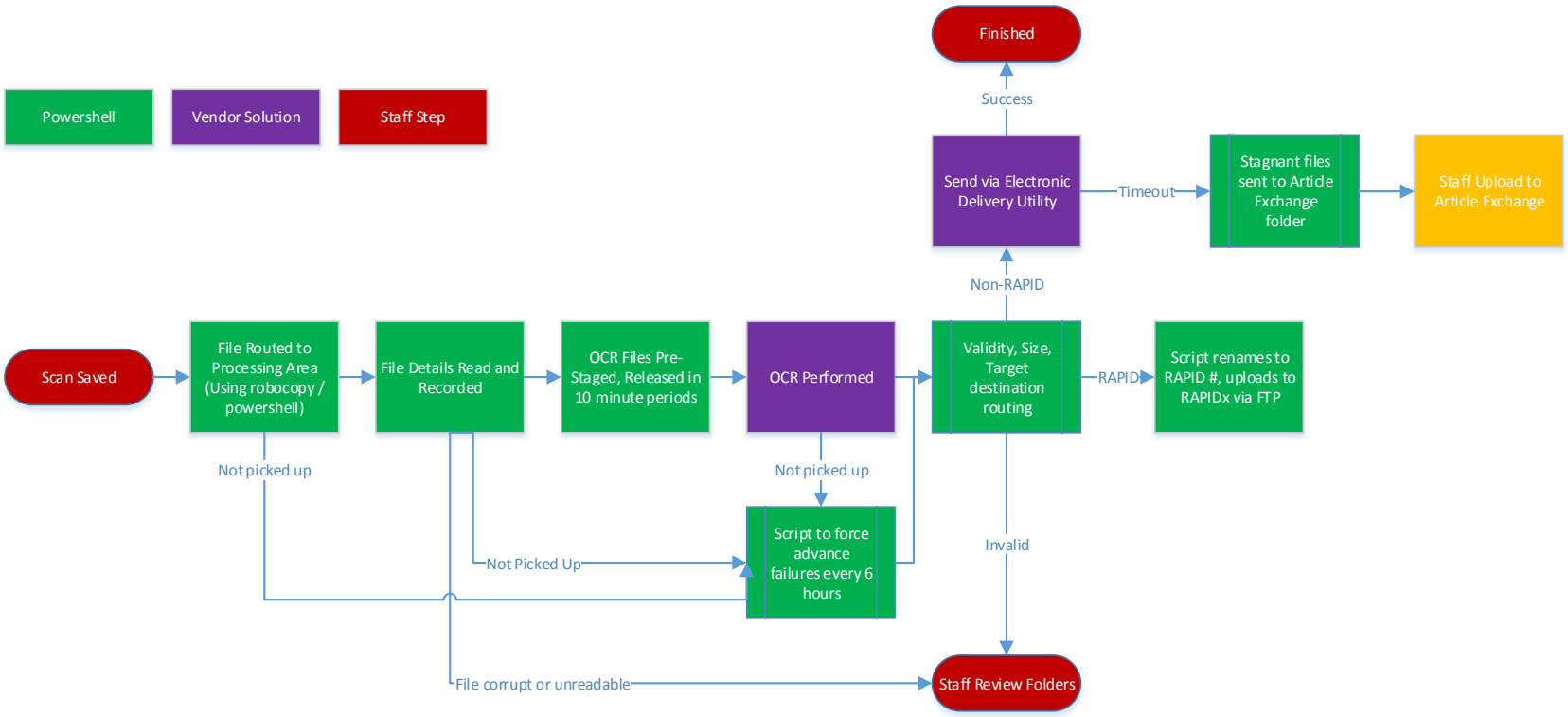
Offsite Scanning Integration – External App

- Needed to make a web endpoint that could be accessed by the GFA LAS (Library Archival System) software used by storage facility
- Needed specific formatting and access rules
- Used an external application due to web access requirement

File Cleanup and Routing – Custom Script

- Consolidated scan unit meant no need to learn ILLiad – just scan the pages on the form and save by TN.
- Needed to filter out various problem transactions
  - Wrong status
  - Corrupt PDF
  - File too large
- Needed to route files based on type to different destinations
  - Split out course reserve transactions
  - Split out RAPID transactions

File Cleanup and Routing – Custom Script



Yale UNIVERSITY LIBRARY

Custom Billing – Custom Script

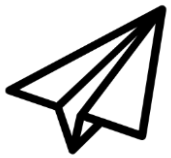
- Centralized physical loan lending means segregated address books can't be used to manage billing
- Modified address book to add lending numbers starting with 100 for special policies – e.g., free reciprocal from a single branch
- Duplicate addresses meant staff needed to assign each time
- Developed script to (1) set default address and (2) identify location codes with different billing rules and apply "special" lender address record
- Developed script to copy over address book from maintained central library list to any libraries that opted to stop maintaining their own address records

Screen Transactions Immediately – Server Add-On

- Needed to evaluate incoming lending requests and weed out those that we can't fill
- We subscribe to IDS Logic which gets availability and shelf location
  - Server add-on
  - Plug-and-play – the addon "bootstraps" itself, automatically downloading the newest code for every run.
  - The more information you give it (especially circulation policies) the better the results
  - Many other features and one of the easiest enhancements to implement

Route Local Transactions Automatically – Add-On

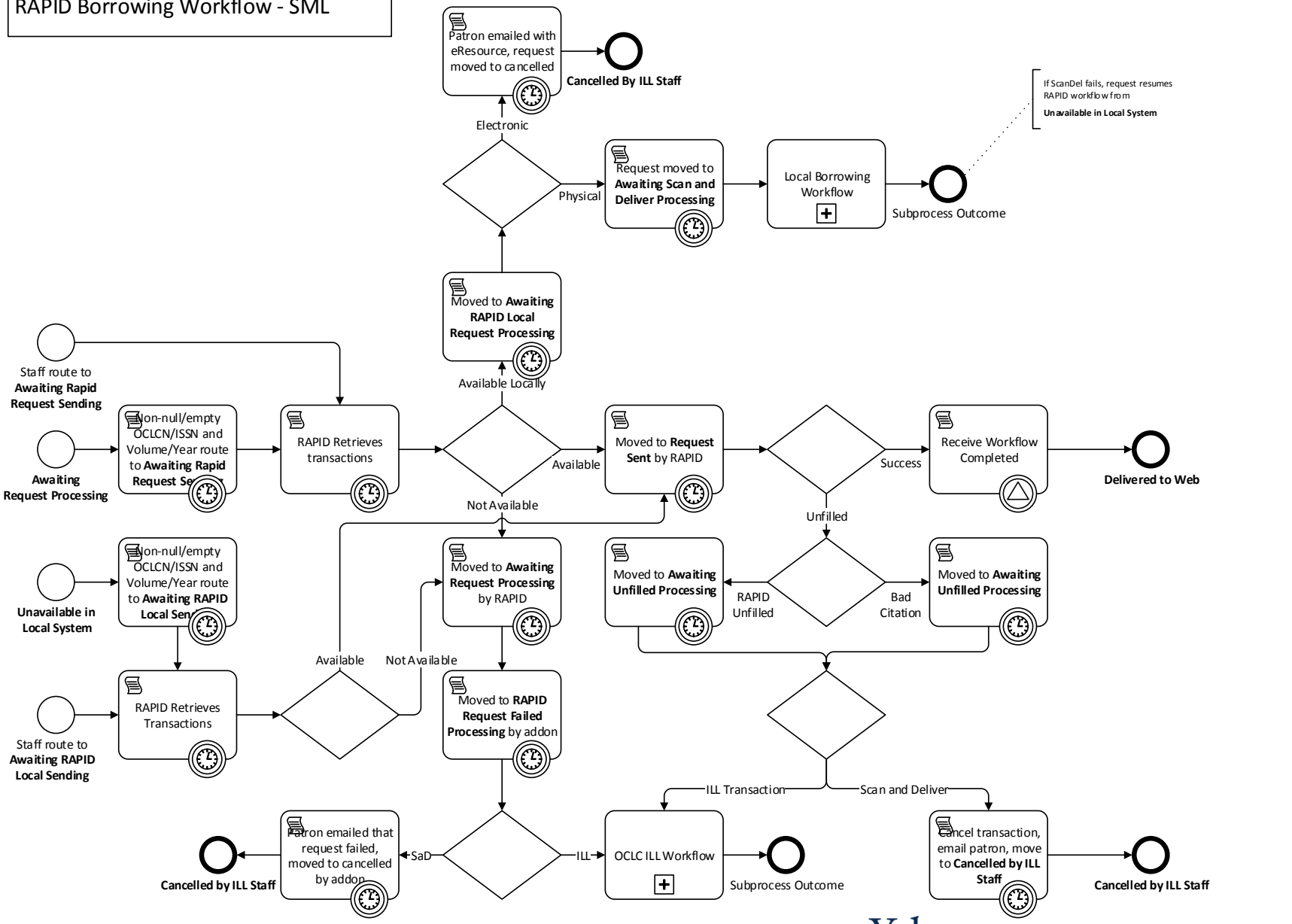
- Local transactions come in that may be located in the central library or one of the branches
- Rather than using the document delivery module all requests are treated as borrowing/lending
- A list was created of all location codes to be expected in OpenURL and then mapped to their owning NVTGCs
- A script updates the lending string and moves transactions to "Awaiting Local Sending"



Move Transactions Into and Out of RAPID – Add-On

- When RAPID identifies an ILL request as held locally it is automatically moved to scan and deliver
- When local transactions can't be found they are automatically sent to RAPID
- When a scan and deliver transaction fails both locally and through RAPID then it is automatically cancelled and the user notified.
- When an ILL transaction fails in RAPID it is sent out to OCLC

RAPID Borrowing Workflow - SML



Yale UNIVERSITY LIBRARY

Things to be careful of

- Programming mistakes
  - They're easy to miss and can be intermittent in effect
  - Mistakes with the email sending tools can cause patrons to repeat notifications every minute
  - Mistakes with routing can leave transactions stuck in a queue that's not watched
- Kafkaesque Borrowing Interactions
  - Avoid any situation where a user may be led to place an ILL request and then have that request cancelled automatically
  - Be mindful of keeping your RAPID holdings up to date
- Assumptions about request data
  - Any patron-input field may contain wildly unexpected information
  - Always perform as much cleaning and checking as possible

Tips for Server Addons

- Use error handling
  - Lua has a robust `pcall()` function which allows you to prevent errors from taking down the whole addon (and your server)
  - Capture and output as much error information as possible. Make sure to empty all inner exceptions but to do it in a protected way (to avoid an error within an error)
- Capitalize on non-Lua tools
  - Atlas has implemented a package called luanet which gives you access to much of the .NET framework – use it!
  - Use the Microsoft Developer's Network to find information about .NET libraries
- Use a tool to simulate server addons
  - Initial development can be very slow while waiting 1 minute for every attempted run of the addon code
  - A server addon simulator can be used to rapidly develop in a similar environment allowing syntax and other simple errors to be caught early.

Tips for Server Addons

- You *can* connect with read/write access to any database, including your own
  - Server addons provide a mechanism to read a .dbcx file just like client addons
  - If you want to make updates to ILLiad you can use this to write to the database – however, if you edit fields inappropriately you risk destabilizing the application
- Write your addons to take actions on a transaction in series
  - E.g., first `RouteTransaction()` then `SendEmail()` then `ESPUpdate()`
  - When you implement functions like these to wrap the built-in Atlas functionality it makes addons *very* easy to reuse – you just update the config files for what transactions you're looking for and you have another addon.

Conclusions

- The new normal includes more services and more requests
- New services can often be channeled through existing workflows
- Analysis first, technology second (at the earliest)
- Choose development type based on resources and what you want to do
- Good addon code is both reusable and reliable
- Have fun – in software, all things are possible!

Referenced Projects will be made available here after the conference:
https://github.com/yalelibrary/

Questions?

Steelsen.Smith@yale.edu
SteelsenS@gmail.com