QUALITY INTELLIGENCE

Intelligence about software quality, intelligently applied

Modeling Scenarios Based on Data

Fiona Charles

CAST 2009

Who I Am:

Fiona Charles - Test Manager and Consultant, President of Quality Intelligence, Inc.

- 30 years in the software development world
- QA & Test Management on software development and integration projects for clients in retail, banking, financial services, health care and telecommunications
- Consulted on testing and test management practices for clients
- Managed E2E Systems Integration Tests on several large programs, including:
  - A retail frequent shopper program
  - An online store for a major retailer
  - A bank teller application
  - Internet television

Today's menu

- Introduction
- The POS test problem
- Models for scenario testing
- Conceptual framework
- Modeling the POS test
- Conclusions

Photo Fiona Charles © 2008, 2009

I hope you'll take away from this session

- New or renewed interest in scenario testing
  - Adding it to your repertoire
  - Enhancing how you do it

- Understanding of a framework for test modeling
  - Value
  - Practical example you can apply for transactional systems
  - Ideas for creating modeling frameworks that fit your own testing problems

Scenario testing is a powerful way to find bugs that matter to stakeholders

- Exercise a system with real sequences of transactions and events
- Find problems in the end-to-end processing that deep functional testing can miss
- Represent various stakeholders' points of view
- Give stakeholders a test they can relate to

What do we mean by "scenario"?

An outline or model of an expected or supposed sequence of events

> The American Heritage® Dictionary of the English Language: Fourth Edition, 2000.

"A scenario is a hypothetical story, used to help a person think through a complex problem or system…A scenario test is a test based on a scenario."

> Cem Kaner, An Introduction to Scenario Testing, 2003

A testing problem

Photo Fiona Charles © 2008, 2009

Does this sound familiar?

- A POS (point of sale) product, customized by the Vendor
- Customer planning to implement POS in 51 stores with a tight timeline
- “Integrator” (my client), responsible for making sure POS is fit for purpose and implemented on time
- Minimal documentation for POS, no pre-delivery access, minimal possibility of contact with development team

And, by the way…

- Strategic project for Customer: staging for POS implementation in 1000+ main stores, with full integration to store and corporate systems
- Vendor doing major customizations for Customer's business processes
- POS reputed to be buggy on delivery
- Customer CIO had prior experience with Vendor
- Integrator on fixed-price contract, small testing budget

Some other testing challenges

- POS runs on locked-down cash registers
  - Testing dependent on specially-equipped test lab
  - Test automation with standard tools not possible
  - No screen capture or session logging software
  - Bug tracker had to run on laptops
- PC version of POS is not identical
  - Minimal benefits from testing before lab ready
- POS operates on a “sales day” principle
  - Must be able to account for every transaction on a given day

Key test strategies (the what)

- Leave detailed functional testing to Vendor POS system experts
- Add test value by supplementing Vendor testing
  - Verify the financial integrity of the end-to-end solution
- Create a test model mirroring Customer's daily business operations
  - Model our test with business scenarios

Key test strategies (the how)

- Apply a conceptual framework, based on data, to construct the test model
- Use a “building block” approach to constructing scenarios, starting from single item transactions
  - Verify solution basics
  - Reuse in simple or complex combinations
- Construct all test cases in spreadsheets, making use of Excel functions where possible

Models for Scenario Testing

Photos Fiona Charles © 2008, 2009

Modeling for scenario testing operates at two principal levels

- The overall model for the test of the system or solution (and its systems context)
- The scenarios that encapsulate the tests to be evaluated

Why "modeling"?

- In every test, we make choices
  - What to include
  - What to leave out

- Consciously modeling a test gives us a way to direct, control, examine, and explain those choices

"Every model is ultimately the expression of one thing…we hope to understand in terms of another that we do understand."

Gerald M. Weinberg, An Introduction to General Systems Thinking

Some useful model types for scenario testing

- Business operations
  - "Day (week, month, mock-year) in the life"
  - Model office or business
- Entity lifecycles or flows
  - Customer experience
  - Lifecycle of a bank account
- Data
  - Static
  - Semi-static
  - Dynamic
- Functional or organizational decomposition
  - Functional areas within the system or business (Ordering, Inventory Management, Billing, etc.)
  - Processes in each area (Order capture, provisioning, etc.)
  - Functions within each process (Enter, edit, cancel order, etc.)

It's a good idea always to combine 2 or more models

- Each model type can act as a source of test ideas
- Different models can act as cross-checks on each other, generating ideas for complementary scenarios, e.g.,
  - Combining the real-world view of a business operations model with the systems view of a model based on the data

A conceptual framework for modeling based on categories of data

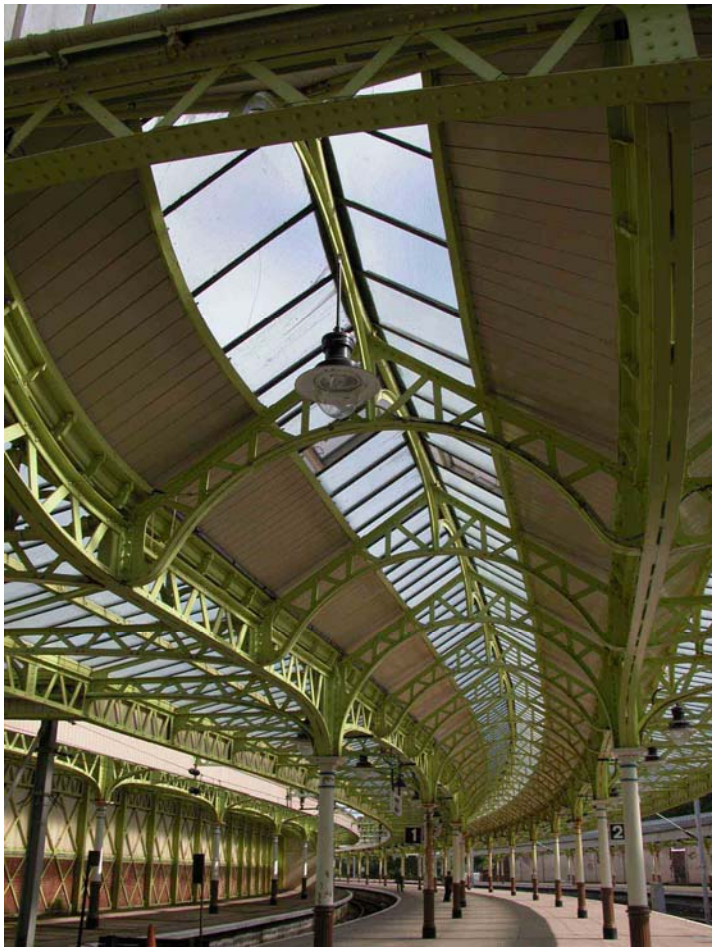


Photo Fiona Charles © 2008, 2009

What does a model framework do?

- Scenarios are stories modeling some aspect of the system
- A conceptual framework gives you a way to structure the stories
  - Analyze the test requirements:
    - powerful tool for driving out variations to feed scenarios
  - Design the test model
- You can create building blocks for reuse in different combinations
- Helps to ensure you don't miss anything important

You can construct a conceptual framework to facilitate developing any model

- Ask “what are the key constituent elements of this model?”
- Create a spreadsheet or other structure to order the elements
- Use the elements to capture variations and apply the model

A data-driven model focuses on the data and data interfaces

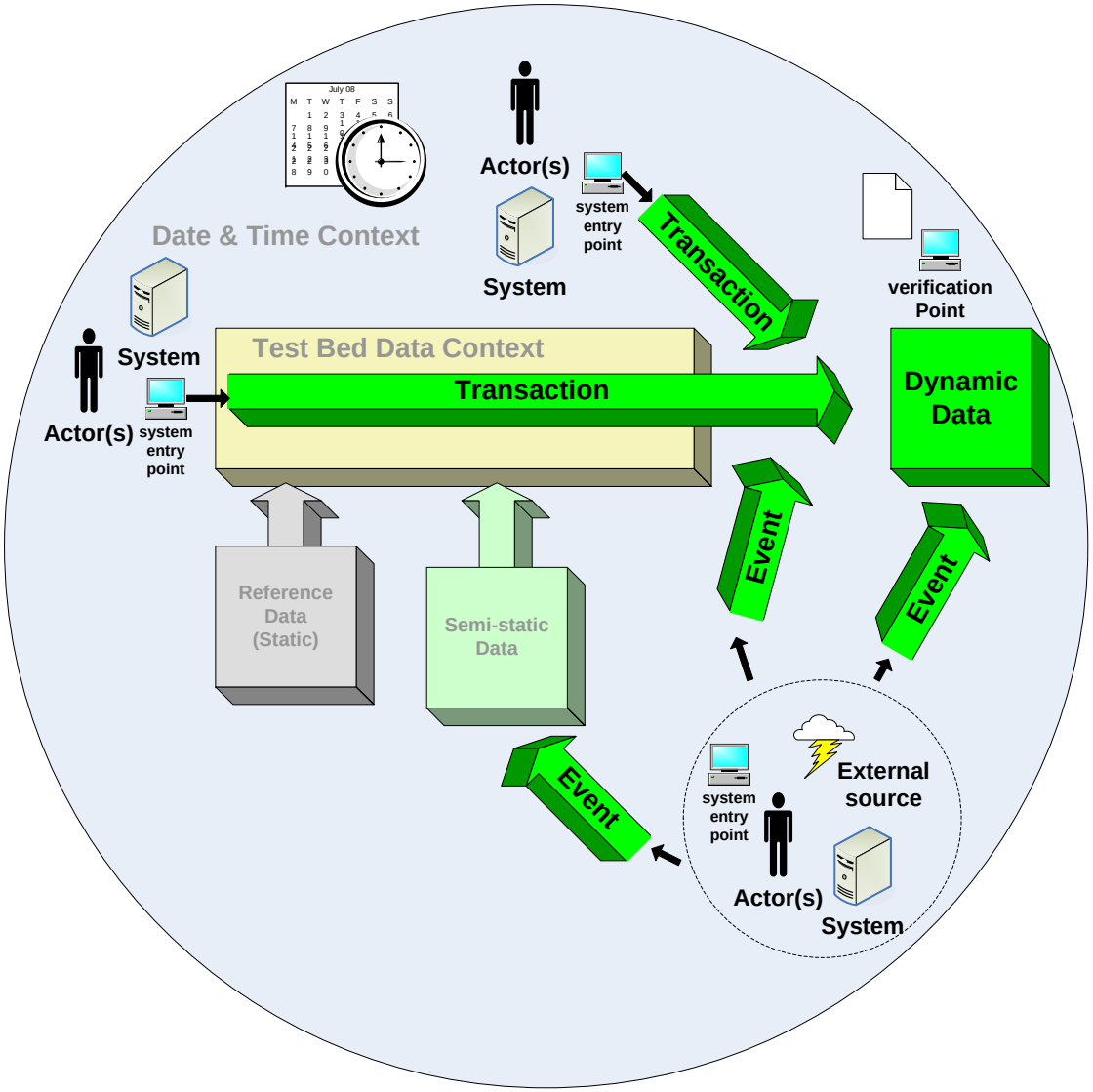
- Inputs, outputs and reference data
- Points of entry to the system or integrated solution
- Frequency of changes to data
- Who or what initiates or changes data (actors)
- Data variations (including actors)

Benefits of a data-driven model

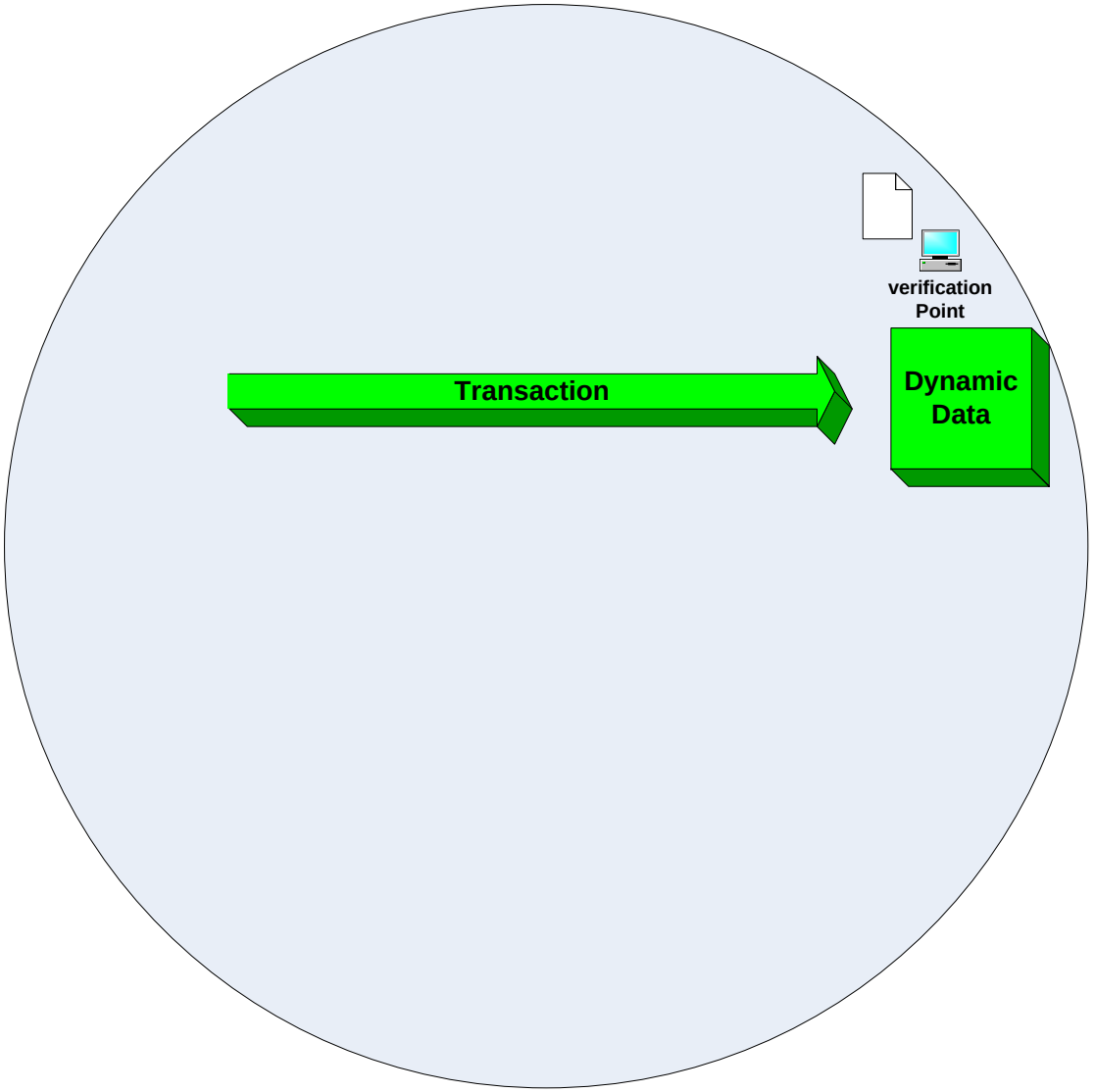
- Represents a systems view that can be rounded out using models based on a business view
- Identifies principal components of the execution model for the test:
  - Setup data
  - Entry points
  - Verification points
  - Events to schedule
- Easy to structure and analyze

A conceptual framework for modeling a scenario test

A high-level framework based on data is useful for modeling a test of a transactional system (or multi-system solution).



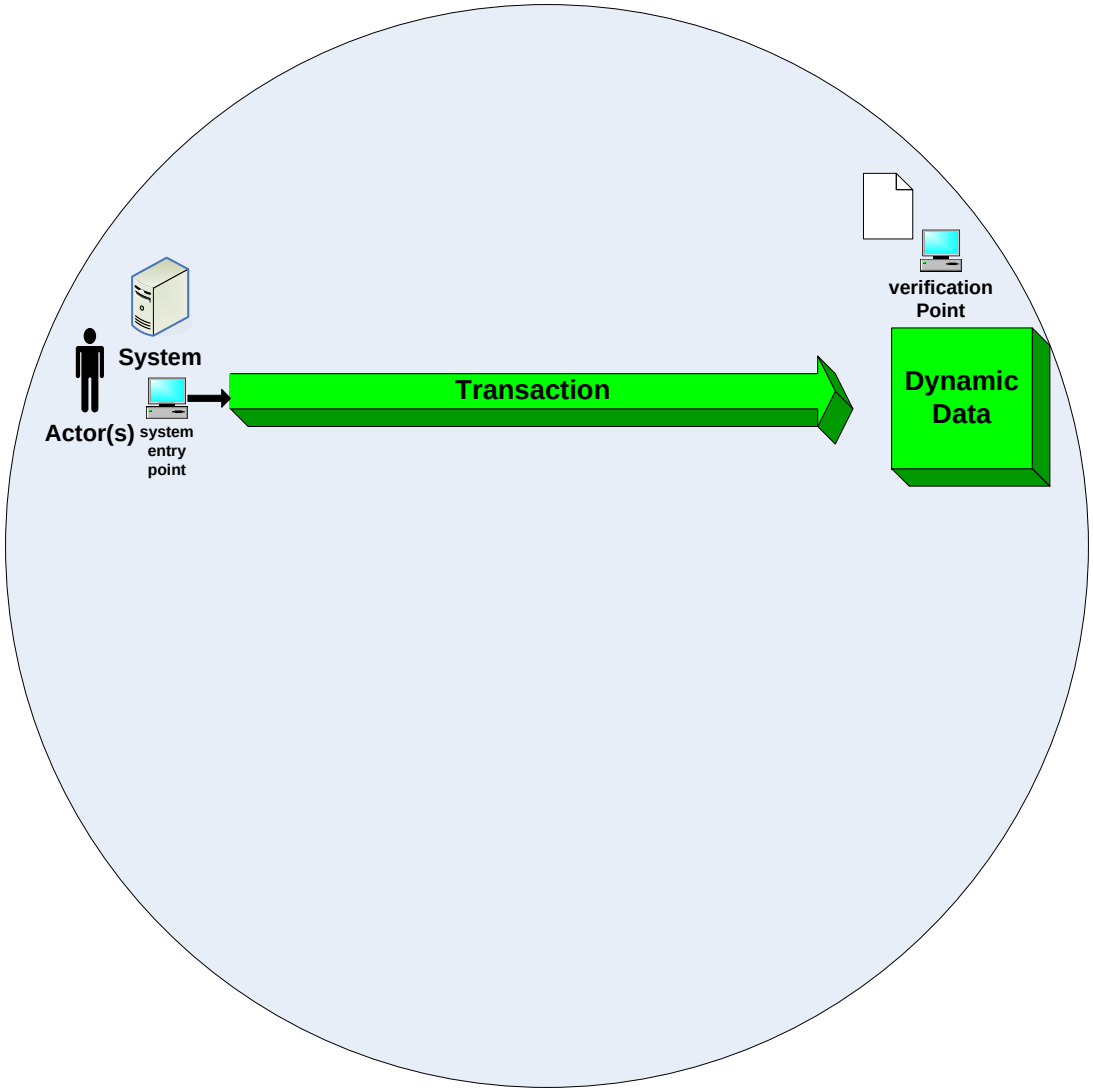
At the most basic level, we want scenarios to test the outcome of system transactions.



Transactions drive dynamic data (i.e., data that changes in the course of the test).

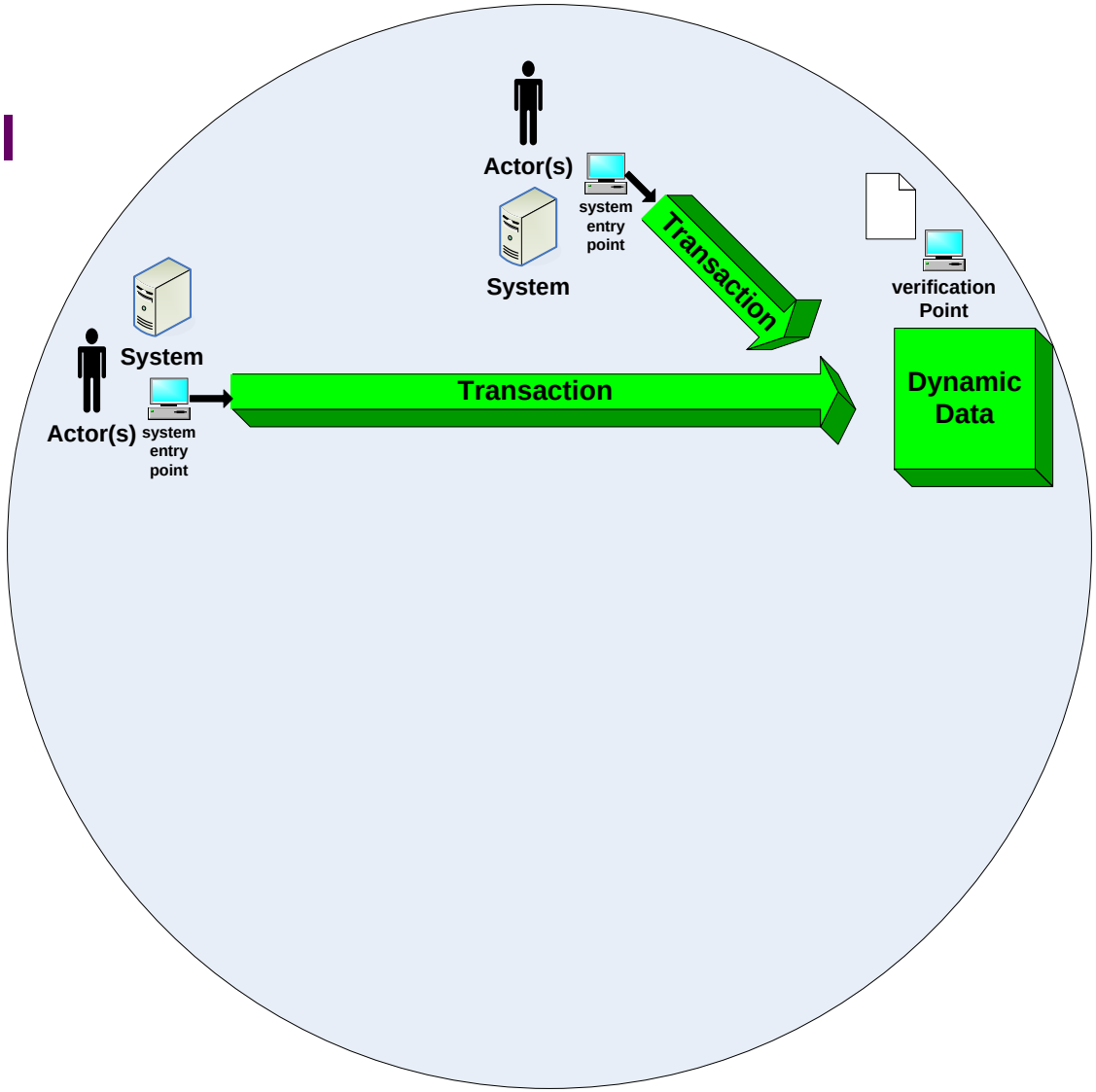
Transactions represent the principal business functions the solution is designed to support, e.g. sales, returns, loyalty point redemptions, customer service adjustments.

A transaction could be initiated by a human actor or by the system.



There may be more than one **actor** involved in a transaction, e.g., a system user and a customer.

Many scenarios will have multiple transactions.

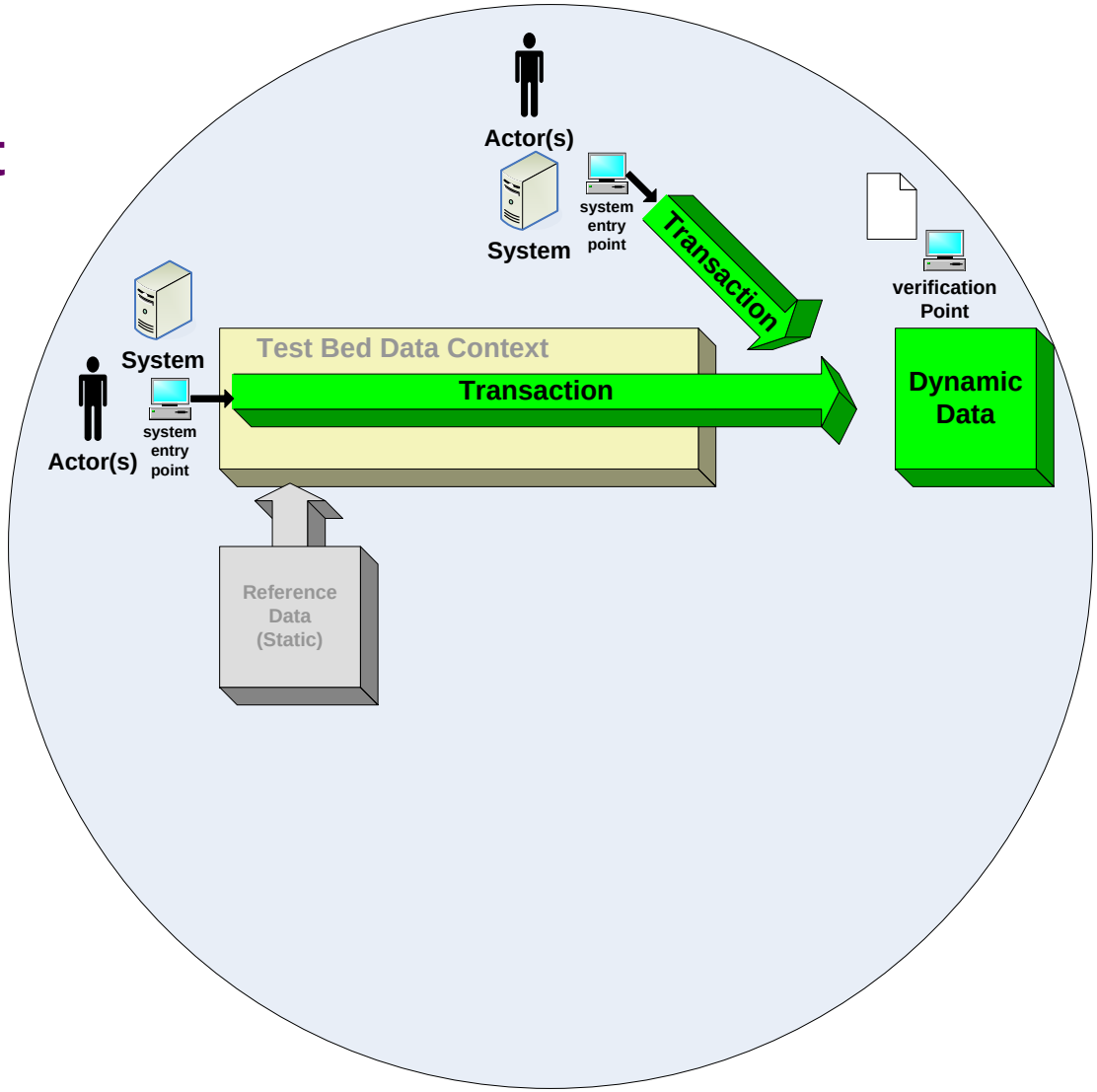


Subsequent **transactions** can affect the outcome by acting on the dynamic data created by earlier related transactions.

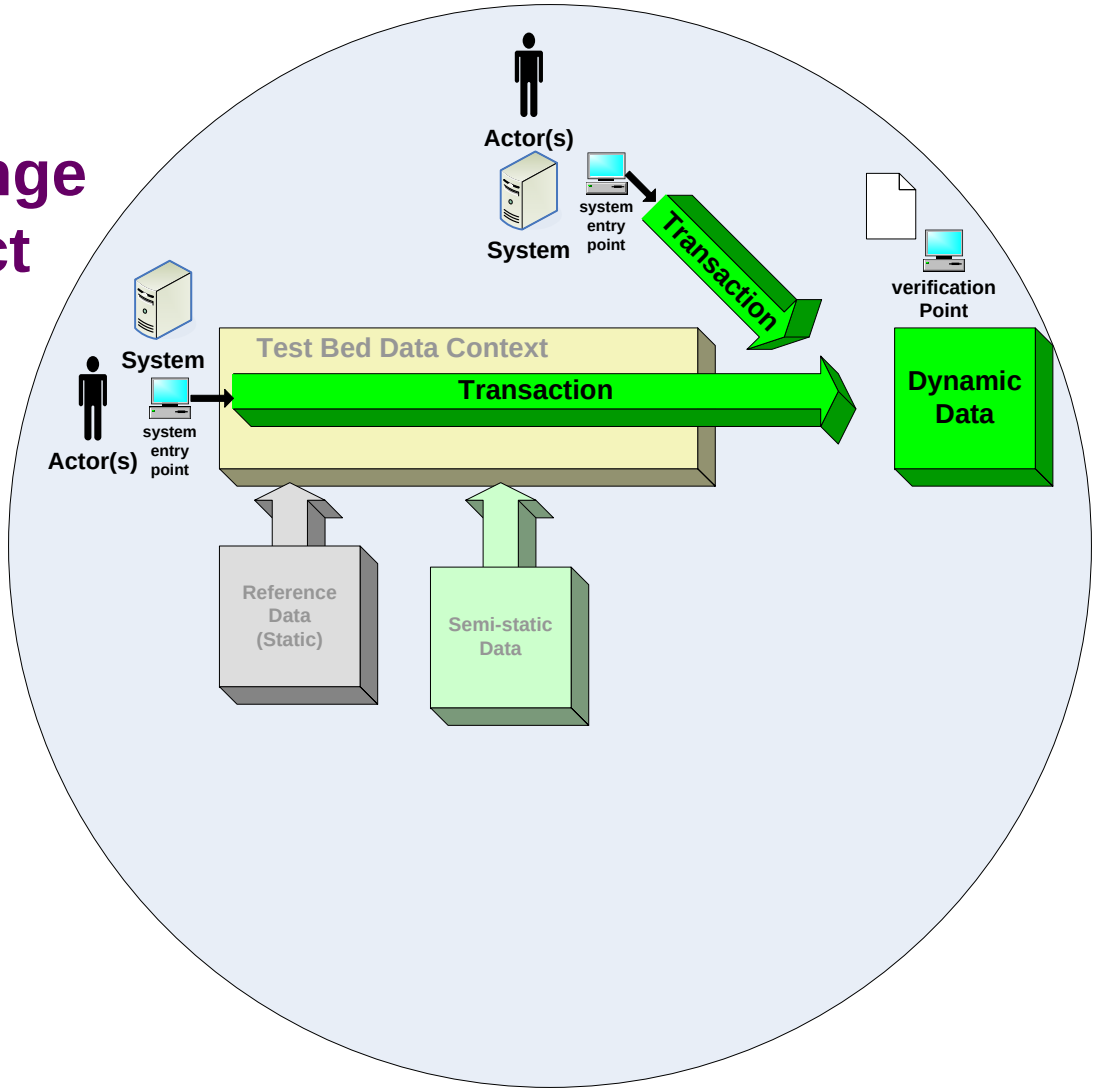
Transactions operate in a context of test bed data

Reference test bed data is static (doesn't change in the course of the test), e.g., user profiles, frequent shopper (loyalty) points awarded per dollar spent, sales tax rates.

Deciding which data should be static is a test strategy decision.



The test bed also contains semi-static data, which can change the context and affect the outcomes of transactions.



Semi-static data changes occasionally during testing, as the result of an event. Examples include items for sale, prices and promotions.

Determining which data will be semi-static, and how frequently it will change, is a test strategy decision**.**

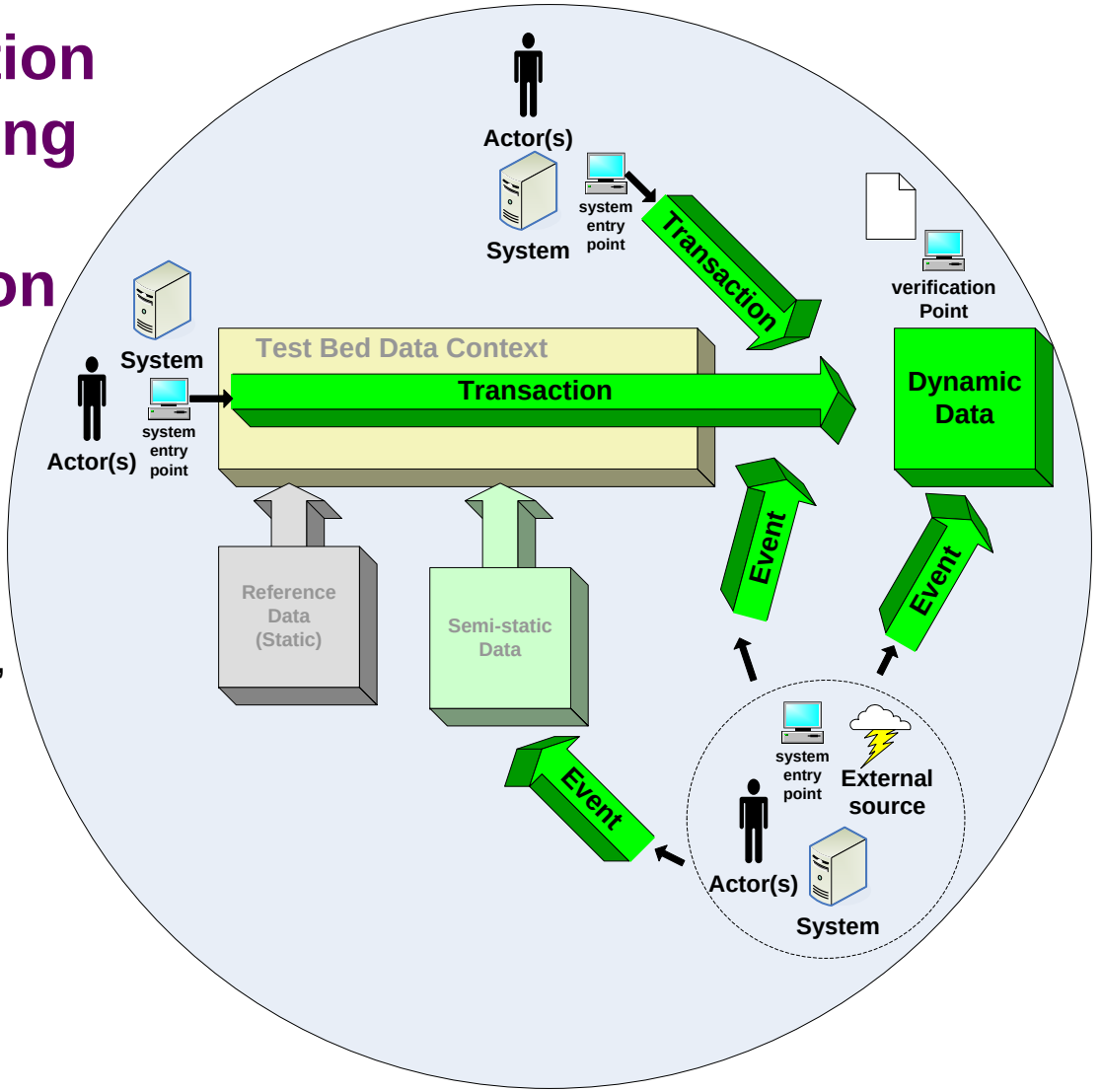
Events affect transaction outcomes—by changing the system or data context, or by acting on the dynamic data

Events can represent:

periodic or occasional **business processes**, e.g., rate changes, price changes, weekly promotions, month-end aggregations

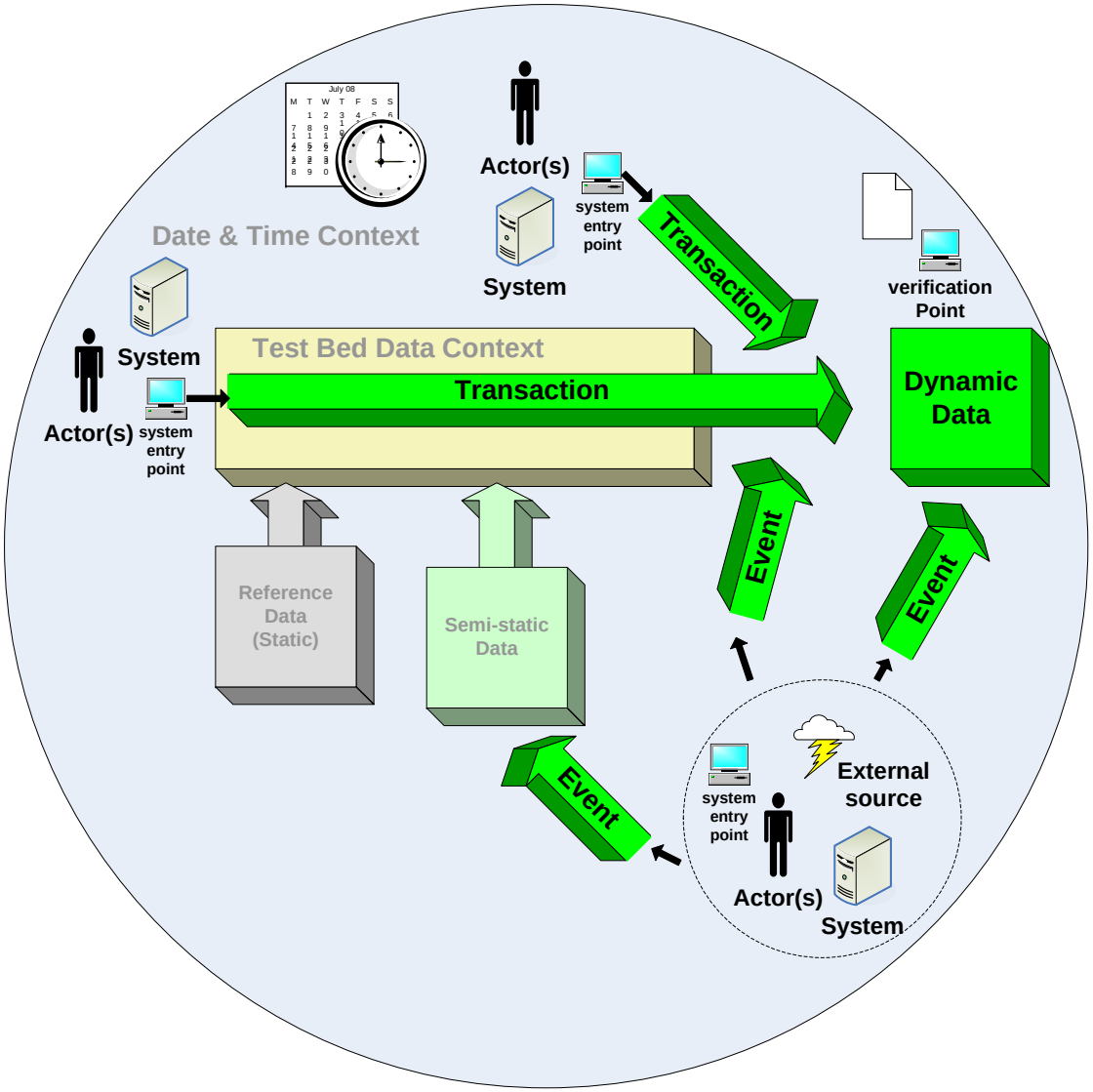
happenings, such as system interface failures

"external" **business exceptions**, such as a shipment arriving damaged, or a truck getting lost



Scenarios also operate within a context of date and time

The **date/time** a transaction or event occurs can have a significant impact on a scenario outcome.



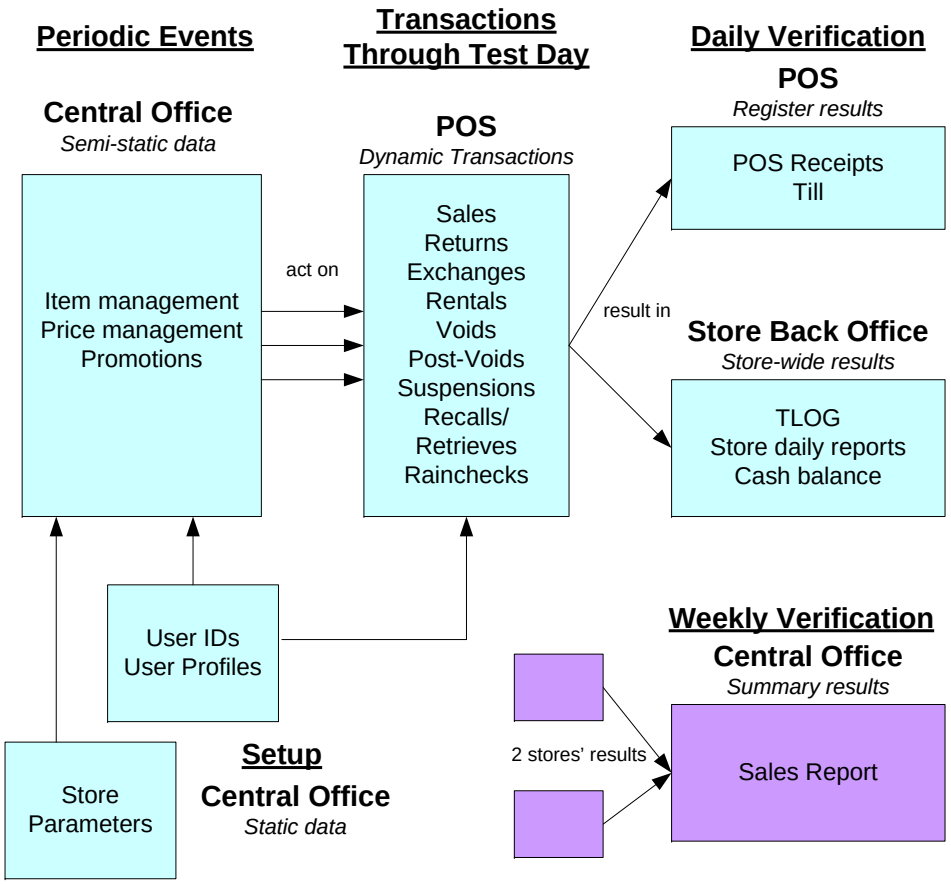
Categorizing the data determines each type's role in the test, and gives us the overall model

- Reference data sets the context for scenarios and their component transactions
- A scenario begins with an event or a transaction
- Transactions have expected results
- Events operate on transactions and affect their results
  - A prior event changes transaction context, e.g., an overnight price change
  - A following event changes the scenario result and potentially affects the transaction, e.g., product not found in the warehouse
- Actors influence expected results
  - User privileges
  - Customer discount or tax status

Modeling the POS test

Photo Fiona Charles © 2008, 2009

Overall model for POS test



We drilled down on the attributes of each framework element, and the possible variations for each attribute

- **Transaction**
  - Type (sale, return, post-void, rental, rental-purchase…)
  - Timeframe (sale/rental + 3 days…)
  - Completion status (completed, void, suspended)
  - Store
  - Register
  - User (cashier, manager, supervisor, store super-user)
  - Customer (walk-in, preferred, loyalty, status native, employee…)
  - Item(s) [multiple attributes, each with its own variations]
  - Tender (cash, check, credit card, debit, coupon, loyalty redemption…)
  - Loyalty points (y/n)
  - Delivery/pickup (cash and carry, home delivery…)
  - Promotions in effect (weekly flyer, store clearance…)
  - Other discount (damage…)

We designed a full range of item transactions in spreadsheets, working through the variations and combining them to make core test cases

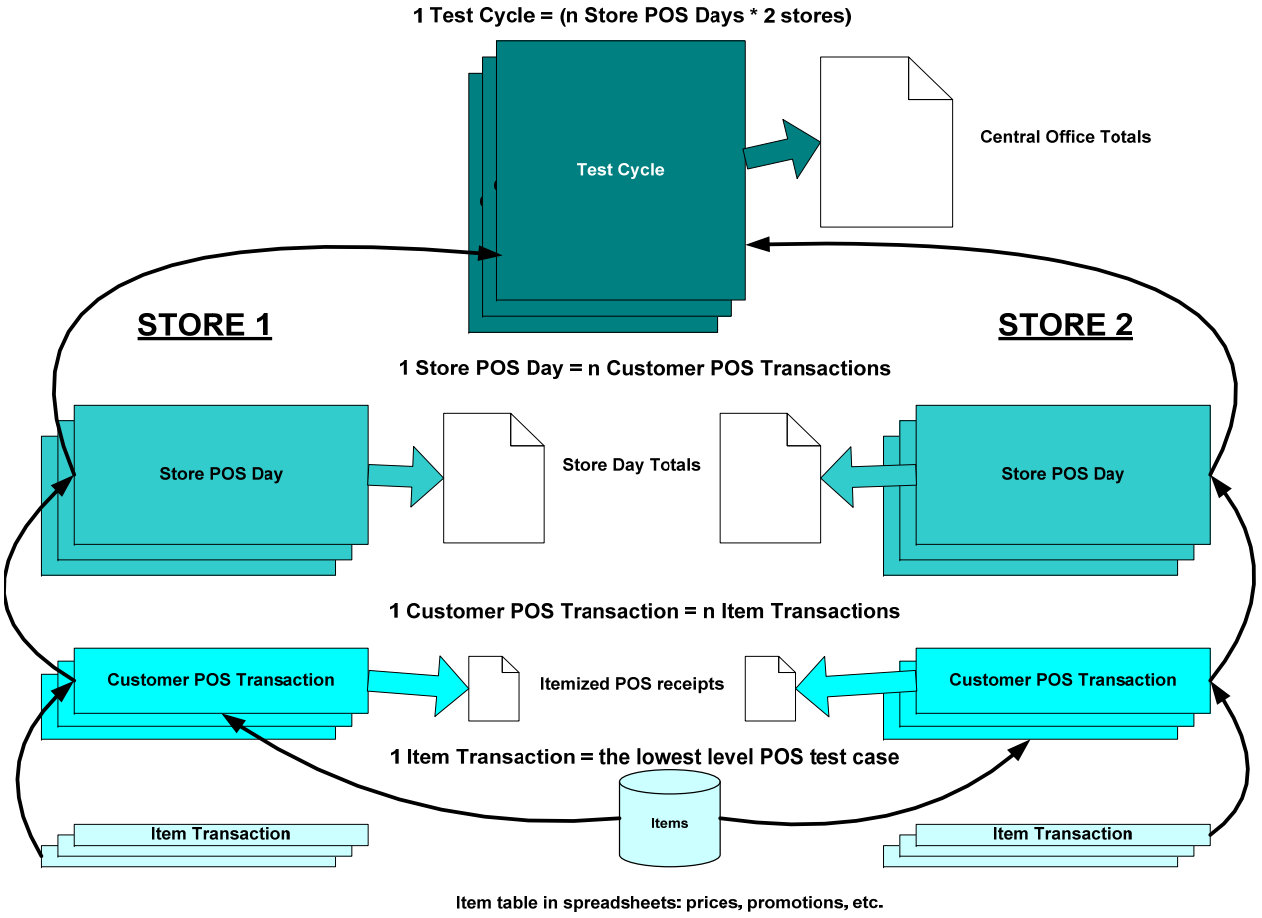
- Shortcuts wherever possible, e.g., Excel lists for variations of most attributes

Transaction design - simplified example:

Txn-ID	Store	Type	Timeframe	Completion	Register	User Profile	Customer	Item(s)	Tender	Points	Delivery /Pickup	Promos	Discounts
POS-1	1	sale	n/a	C	4	cashier	walk-in	45270	cash-CAD	n	C&C	n/a	n/a
POS-2	1	return	sale+1	C	2	manager	walk-in	45270	cash-CAD		n/a	n/a	n/a
POS-3	2	sale	n/a	c	1	cashier	employee	98651 54945 21498	Visa	y	HD	sidewalk	raincheck

We designed events, including price changes and promotions, in a similar fashion

We designed scenarios using the item transactions as the lowest-level building blocks

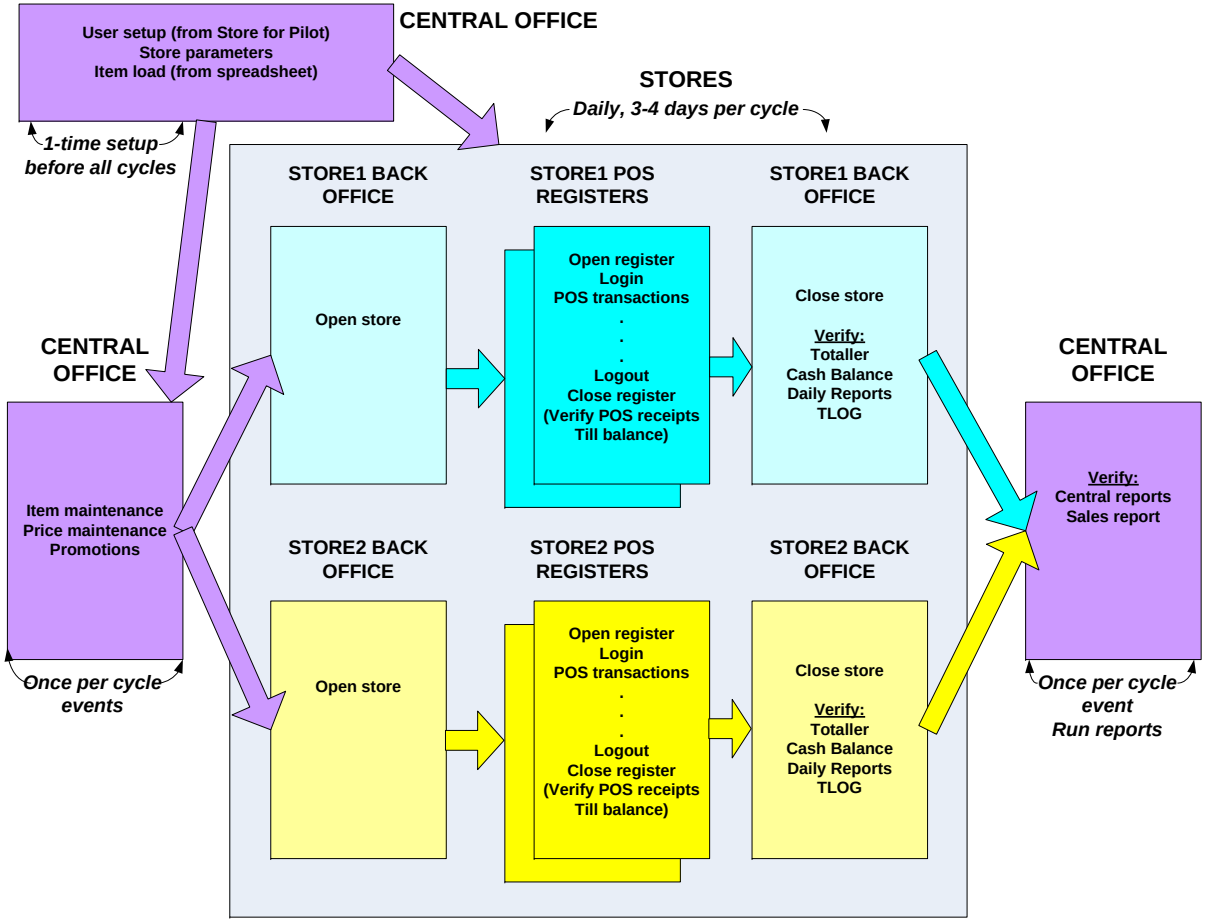


Scenario designs included events

Scenario design - simplified example:

Scenario ID	Description	Prior txns	Prior events			Main Txn	Follow-on Txns	Post-Txn Events
			Central Office	Store Back Office	Store			
POS-S-S1	Senior phones in to buy 3 items on Super Senior's Day of which 1 is a charge sale item and 1 is a govt funded item, delivery and service charges applied, pre-paid delivery	n/a	PROMO-45 PROMO-7	Open Store	Open Register Login	POS-S133		Logout Close Register Close Store RPT-16
POS-S-S19	Loyalty customer pays for 5 items, of which 4 are points-eligible; customer changes mind before leaving register and 1 points-eligible and 1 non-eligible are post-voided	n/a	n/a	Open Store	Open Register Login	POS-S17	POS-PV17	
POS-S-S65	Senior phones in to buy 3 items on Super Senior's Day of which 1 is a charge sale item and 1 is a govt funded item, delivery and service charges applied, pre-paid delivery	n/a	PROMO-12 PROMO-7 ITEM-96	Open Store	Open Register Login	POS-S133	n/a	`

Test execution cycle model



Spreadsheet test artifacts, for flexibility and ease of use:

- Standard structures
- Built-in calculations for expected results
- "Building block" concept, with components shared among artifacts
  - External references
  - Background tables for test bed data
- Macros to automate assembly/recreation of artifacts from others

An item transaction sheet pulled in data values for test execution

Test Case #: POS-3901

Customer Scenario: Sale

Organization Txn # / Store # /Register: 3967 / 8001 / 1001

Register 1

Run details: SIT Test - Cycle28b Day 1, V1.24k

Customer Profile:

Customer	Customer Address	Shipping Address	Phone #	Credit Card / Tender Information: Type	Credit Card / Tender Information: Number	Loyalty Program and Account #	Total Points
6 - Shopper Six Oak	266 VILMA DR OAKVILLE, ON L6L 3J8	Customer-Carry-out	905-111-2249	Visa 2	XXXX XXXX XXXX 0855	Loyalty Reg #N/A	#N/A

Multiple Tendering

Cash	Debit	Visa	Mastercard	Amex	USD	USD-Exch	G-Cheque	O-Cheque	Store Chq	Trav Cheque	Gift Card	COD	Charge			
		Y														

Sales Person / BadgeID / Store: Store Cashier / 8209 / Pleasantville

Selling Store: 8201 - Del Rio Plaza 3421 King St.

Rental Return Date:

Return Store: -

Issue Rainchecks:

UPC	QTY	Date	Description

ON	Prov. Tax:	1 GOOD	
N	Employee:	N	Preferred:
N	Senior:	N	Tax Exempt:
N	Delivery:	N	Recall
N	Post Void:	N	Suspend

Store Transaction Details:

Ln #	UPC	Model / Serial / Asset Tag	Description	Quantity	Pack Sold	Pack Quantity	Enter after price prompt	Manual Price Correction (PriceBand red)	Unit Price LookUp	Line Price (with overrides)	Promo / Discount Apply Before Tax	Promo / Discount Apply After Tax	Final Price	Item Eligible for Loyalty Points	Loyalty Eligible Dollar Amount	Promo Bonus Points	Sales Tax	PST	GST	HST	Sales Dept	Retail Dept	Category	Fine Line	Item Type	Allow Return	Price Band	Line Item Void	Line Total	Issue Coupon
1	57565182009	018-200	SMOOTH EXAM PAPER	1	Pack	12			$46.99	$ 46.99	$ 14.10	$ 0.46	$ 32.43	Y	$ 32.43	#N/A	ONPT	$ 2.63	$ -	$ -	HL	DA	271	257	PACK	0	1	N	$ 35.06	$100 Voucher
2	38472360513	38-307-01	POLYSORB WALK/RUN IN	1					$29.99	$ 29.99	$ 18.00	$ 0.31	$ 11.68	Y	$ 11.68	#N/A	ONGT	$ -	$ 0.72	$ -	HS	FC	259	572	Standard	0		N	$ 12.40	
3	POS-1901	RD01-US	RD01-US	1					$699.99	$ 699.99	$ 210.00	$ 16.88	$ 473.11	Y	$ 473.11	#N/A	ONGP	$ 39.20	$ 29.40	$ -	HL	BS	272	747	Standard	0		N	$ 541.71	
4	627394930036	DL930CA	10,000 LUX BRIGHT LIGHT	1				$ 239.99	$249.99	$ 239.99	$ 84.00	$ 2.35	$ 153.64	Y	$ 153.64	#N/A	ONGP	$ 12.48	$ 9.36	$ -	HD	DT	267	224	Serial	0		N	$ 175.48	
															$ -													N		
															$ -													N		
															$ -													N		
															$ -													N		
															$ -													N		
															$ -													N		
															$ -													N		
															$ -													N		
															$ -													N		
															$ -													N		
															$ -													N		
Total				4					$1,026.96	$1,016.96	$ 326.10	$ 20.00	$ 670.86		$ 670.86	#N/A		$ 54.31	$ 39.48	$ -									$ 764.65	

Promo / Discounts - Work Area:

Ln #	UPC	Quantity or Rental Periods	Price Override (Before Tax)	Price Correction (Before Tax)	Unit Price (with overrides) (Before Tax)	Line Price (with overrides) (Before Tax)	Item Minimum Price (Before Tax)	Employee 30% (Before Tax)	Senior 20% (Before Tax)	Flyer Promo % (Before Tax)	Flyer Promo $ (Before Tax)	Promo ID	Raincheck (Before Tax)	Store Discretion $ (Before Tax)	Discretion Reason	Store Discretion % (Before Tax)	Discretion Reason	Referral Coupon % (Before Tax)	Lowest Price Free (Before Tax)	Vendor Coupon $ (After Tax)	Vendor Coupon % (After Tax)	Store Coupon (After Tax)	Gift Voucher (After Tax)	Store Gift Voucher (After Tax)	Coupon Serial #	3rd Party Discount $ (After Tax)	Manual Adj. (Before Tax)	Manual Adj. (After Tax)	Rental Rebate $ (Before Tax)	Promo / Discount Apply Before Tax (Before Tax)	Promo / Discount Apply After Tax (After Tax)
1	57565182009	1			$ 46.99	$ 46.99	$ -	$0.00	$0.00	0%	$ 14.10	SIT-CO-041&042&45													CIBC	$ 0.46				$ 14.10	$ 0.46
2	38472360513	1			$ 29.99	$ 29.99	$ -	$0.00	$0.00	0%	$ 18.00	SIT-CO-041&042&45													CIBC	$ 0.31				$ 18.00	$ 0.31
3	POS-1901	1			$ 699.99	$ 699.99	$ -	$0.00	$0.00	0%	$ 130.00	SIT-CO-037&041&045		$80.00	Price Match					$ 10.00					CIBC	$ 6.88				$ 210.00	$ 16.88
4	627394930036	1		$239.99	$ 239.99	$ 239.99	$ -	$0.00	$0.00	0%	$ 84.00	SIT-CO-037&041&45													CIBC	$ 2.35				$ 84.00	$ 2.35
Total		4	$ -	$ 239.99	$1,016.96	$1,016.96		$0.00	$0.00		$ 246.10		$ -	$ 80.00						$ 10.00	$ -	$ -	$ -	$ -		$ 10.00	$ -	$ -		$ 326.10	$ 20.00

Conclusions

Photo Fiona Charles © 2008, 2009

POS scenario test results

- Test originally planned for 3 cycles (+ regression) over 4 weeks, actually ran through 28 cycles over 14 weeks, and continued after pilot implementation
- Logged 478 bugs, all but 20 of which our Customer insisted on having fixed:

Severity	Count	%
1	8	**2%**
2	331	68%
3	116	25%
4	23	5%
Total	**478**	

Was this the most efficient way to find those bugs?

- Many should have been found and fixed by the Vendor before we got the system
- Others would probably not have been found by any other kind of test
- And—given the total picture of the project—it was critical that the Integrator conduct an independent test, and this was the most efficient way for us to do that with the constraints we had

Business benefits of POS test strategy

- The most efficient way for us to get assurance end-to-end
  - Addressed the needs of both System Integration Test and Acceptance Test
- Finance buy-in
  - Our test verified financial integrity across the system, including interactions between promotions and POS transactions
  - Provided assurance to Finance of solid solution from their point of view
    - Acceptance for implementation in secondary LOB stores
    - Useful for main store LOB decision
    - Solid evidence of testing performed if ever required for future audits
- Provided cross-system business point of view
  - Tested how the solution deals with complex data interactions that are business norms
    - Date-dependent prices
    - Date-dependent promotions
    - Returns when prices have changed, etc.
- Verified critical reports for business-like data

Testing benefits

- Supplemented Vendor testing rather than duplicating
  - Vendor test cases didn't demonstrate integrated view
  - Our test focused primarily on outcomes, only secondarily on immediate experiences
- Building-block approach added efficiency in preparation and execution
  - Starting with item transactions, combined into scenarios and day plans, with referenced item table in background
  - Gave us flexibility to reschedule testing according to state of the system
  - Allowed drilling down to item level (promo setup, etc.) to find a problem
- Provided reusable regression test for Customer
  - Robust transaction-scenario structure would be reusable for future releases, upgrades to infrastructure, etc.
- We could layer Operability Tests on top, simulating real conditions and verifying outcomes

Situations where you should consider scenario testing

- As part of an overall test strategy that includes different kinds of tests
- For Acceptance tests of business systems
  - UAT
  - Vendor acceptance
- End-to-end systems integration tests
- When you don't have detailed knowledge to do under-the-covers testing and have no way to get it
  - Inadequate documentation
  - Restricted access to people who wrote the software
  - Time pressures

Critical success factors for scenario testing

- Domain knowledge
  - Testers with experience in the domain
  - Business input and scenario review
  - Industry books (Cem Kaner's suggestion)
- A model with a framework that fits the type of application
  - A data-driven model works well for transactional systems
  - For other types of applications, e.g., a desktop publishing system, you would need to create a different model and framework, such as one based on usage patterns
- Structured analysis
- Building-block concept: design and build varied and complex tests from elements
  - Begin testing with the simplest cases before adding complexity

Scenario testing risks

- Miss important bugs you could find with under-the-covers or deep function testing
- Difficult to diagnose bugs, especially if you go too complex too soon
- Choosing a model that is too restrictive
- Choosing a model that is too expansive (and expensive)
- Falling in love with one model type and missing the benefits of others

Further reading

Photo Fiona Charles © 2008, 2009

Cem Kaner, *An Introduction To Scenario Testing*; Florida Tech, 2003

http://www.kaner.com/pdfs/ScenarioIntroVer4.pdf

Hans Buwalda, *Soap Opera Testing*; Better Software, February 2004

http://www.stickyminds.com

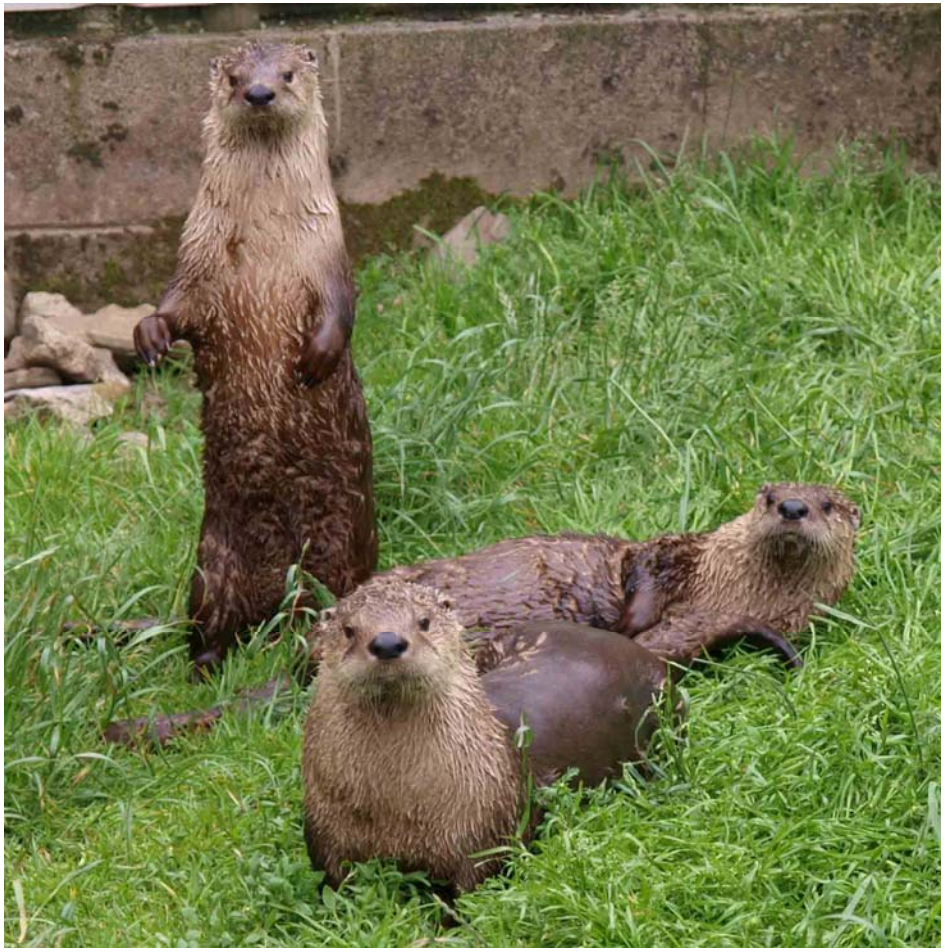


Photo Fiona Charles © 2008, 2009

?

Wrap-up questions and discussion