SOFTWARE DEVELOPMENT STANDARDS

PURPOSE

The purpose of this document is to support and outline in detail the requirements of the Software Development Policy.

Effective development processes are critical to the success of projects. This Software Development Standard provides:

- Development standards for all stages of the System Development Life Cycle
- Minimum requirements for software development activities, deliverables and acceptance sign-off.

Scope

These requirements are mandatory and must be adhered to by all employees (i.e., faculty, staff), consultants and / or contractors involved in the development or modification of mission critical applications that support Brock University at an enterprise level.

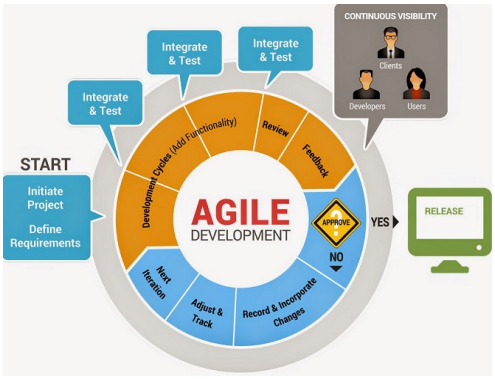
Requirement levels

The following wording conventions are used in this document.

Term	Meaning
Must	This requirement is mandatory, it is not optional.
May	If there are options provided, the implementer is able to choose one or more of the options outlined. At least 1 option **must** be selected.
Should	If business rules countermand a standard practice, deviating from the standard **must** be approved by management as a modification to the standard practice.

Definitions, Acronyms and Abbreviations

Agile Method:
A software development method. Most agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames that typically last from one to four weeks. Every iteration involves a cross-functional team working in all steps: planning, requirements/analysis, design, coding and testing. At the end of the iteration, a working product is demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly. Iteration might not add enough functionality to be useable on its own, but the goal is to have an available release at the end of each iteration. Multiple iterations might be required to release a product or new features.



AODA:
Accessibility for Ontarians with Disabilities Act – legislation.

Application:
Computer programs, procedures, rules and associated documentation and data pertaining to the operation of a computer system.

Audit or Review (Peer Reviews):
An independent review to assess compliance with software requirements, specifications, baselines, standards, procedures, instructions and code.

Baseline:
A specification or end-product that has been formally reviewed and agreed upon. This becomes the basis for further development and must go through change control procedures to be altered.

BSA:
Business System Analyst.

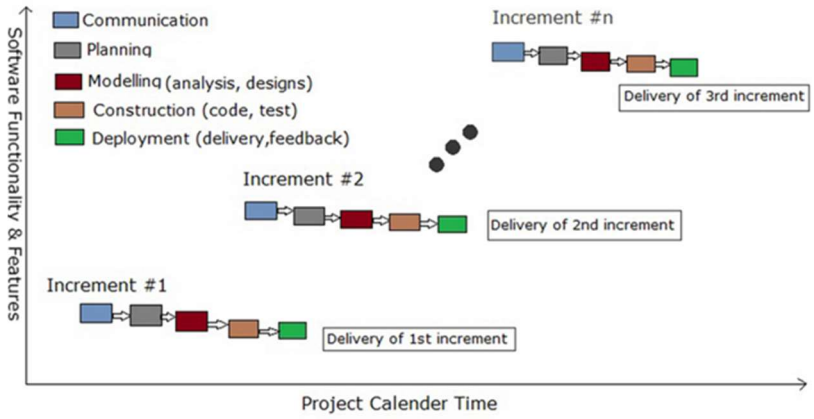
CAB (Change Advisory Board):
A cross functional group representing the various areas in ITS. This group reviews proposed changes to infrastructure, software, networking, firewall rules, etc.

Cross-functional team:
A group of people with different functional and technical expertise working together to achieve a common goal.

DBA:
Database Administrator

Evaluation:
A technique in which requirements, design, code and test results are examined in detail by a person or group to detect potential problems. The results are documented.

Incremental / Iterative Methods:
Software development methods. This method breaks the project requirements into incremental changes (phases or sprints). The series of changes/releases is referred to as "increments", with each increment providing more functionality to the customers. After the first increment, a core product is delivered, which can already be used by the customer. Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly.



Maintenance:
To repair, change or enhance software/application.

Mission Critical:
A system or application whose failure will result in the failure of University operations.

MoSCoW method:
A technique used to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement. Also called the MoSCow prioritization or analysis.

Letter	Meaning	Description
M	Must	The requirement MUST be satisfied in the final solution for the final solution to be considered a success.
S	Should	A high priority item that should be included in the solution if possible. Often a critical requirement that may be satisfied in other ways if strictly necessary.
C	Could	The requirement is considered desirable but not necessary. It should be included in the solution if time and resources permit.
W	Won't	A requirement that stakeholders have agreed will not be implemented in a release but may be considered for the future.

Product:
A product is the tangible result of any process or work group. This includes (but is not limited to) purchased software products, components, code, services and deliverables.

Project Team:
A group of people (may include various departments) who collaborate and work together to deliver a software product.

Regression Testing:
The process of testing changes to software to ensure the changes have not adversely affected current system functionality.

Sign-off:
The declaration that the product has met expectations and been accepted by the governing body of the project.

Software Development Lifecycle (SDLC):
A systematic approach to creating software applications. This cycle typically includes the following seven phases:

1. Requirements gathering - collection of business requirements / needs
2. Analysis - Business and requirements analysis
3. Design - Architecture and application design
4. Coding – Development/programming
5. Testing – Quality assurance, bug fixes, error correction
6. Implementation – Deploying (releasing) the application into the production environment for business use
7. Post Implementation – maintenance and review.

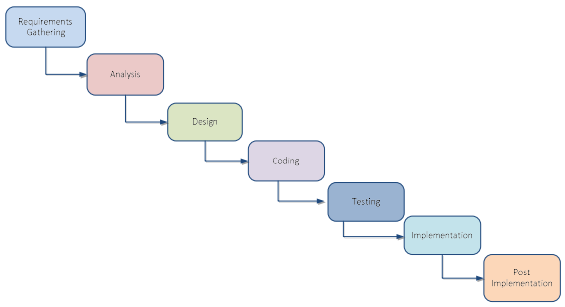
System:
A set of programs which perform all functionality defined within a software application.

Subsystem:
A functionally related to a subset of the system.

User Acceptance Testing (UAT):
UAT testing is performed by the functional business groups that will be using the software in production. It is performed on a test/ quality assurance site that is separate from the production environment.

Walkthrough:
A review process in which an individual(s) leads their peers through their work product. This is used to evaluate requirements, specifications, code, documentation, etc.

Waterfall Method:
A software development method. Waterfall is a sequential design process. Each phase in the lifecycle must be fully completed, documented and agreed upon prior to moving forward to the next phase.



References and Related Documents

Version	Title	Document Location	Date Accessed dd/mm/yyyy
2.1	Standards for dotNet	\\campus.brocku.local\carddfs\develop\documentation\ standards and checklists	10/23/2014
1.0	WCAG2Checklist.pdf	\\campus.brocku.local\carddfs\develop\documentation\ standards and checklists	10/23/2014
	SQL templates for stored procedure creation	\\campus.brocku.local\carddfs\develop\SQL Queries\ TEMPLATES	11/11/2014
1.2	Production Emergency Release to BrockDB	\\campus.brocku.local\carddfs\develop\documentation\ standards and checklists	11/11/2014
1.0	Requirements Template	\\campus.brocku.local\carddfs\develop\documentation\ standards and checklists	11/11/2014

Versioning and/ or Change Management

Requested revisions to these Standards are to be submitted to the Application Architect for approval.

General Standards

All software development projects, including maintenance projects, must follow these standards.

Objectives for application development include:

- Clear definition of purpose
- Simplicity of use
- Ruggedness (difficult to misuse, kind to errors encountered)
- Delivered on time and when needed
- Reliability
- Efficiency (fast enough for the purpose it was created)
- Minimum development cost
- Conform to standards
- Clear, accurate and precise user documentation
- Clear, accurate and precise technical documentation.

All production systems must have designated Owners and Custodians for the critical information they process in order to identify requirements and verify the final deliverables with signoff.

There must be a separation between the production, development and test environments. This will ensure that security is rigorously maintained for the production system,

while the development and test environments can maximize productivity with fewer security restrictions. Where these distinctions have been established, development and test staff must not be permitted to have access to production systems.

All applications are reviewed at predetermined checkpoints of the SDLC by the Application Architect or their designate. Any deviations are reported and corrective action is determined prior to the application being released to production.

Electronic authorization indicating Standards have been met is required before a new or modified application can be released to production.

Throughout the entire project, special consideration must be given on an ongoing basis to capturing and implementing security and privacy requirements. The post-implementation review must reflect this.

Requirements Gathering

Document a clear statement of the current situation outlining problems and opportunities the software will address.

Review existing systems/processes.

Identify and document the issues with the current data/ system/ process.

Document:

- New business needs
- New assumptions since initial project assessment
- Items that will not be solved with this software (outstanding issues)
- Integration points with other software
- Data storage requirements
- Data retention requirements
- Cross-functional support/input required to proceed with this project/request
- Legislative/contractual/security/privacy/access requirements
- Confidential data, access rights to this data and compliance requirements for this data (e.g., payment requirements must meet PCI compliance; student home address must adhere to FIPPA rules, etc.)
- Roles required for security.

- Logging requirements (i.e., what needs to be captured in audit logs other than who updated, when updated, etc.).
- Reporting requirements.
- Training requirements.

Walkthrough and review of requirements:

- Walkthrough with client
- Walkthrough with peer.

Obtain sign-off and approval from client to ensure there is agreement on the requirements as documented.

Deliverables:

- Requirements document.

Analysis

Evaluate the documented requirements. During the evaluation process the following criteria must be considered and results of the evaluation documented:

- Requirement can be traced to business need
- Requirement is consistent with business need
- Testability of the requirements
- Can the requirement be implemented given the current architecture?

Establish and document software requirements:

- Functional and capability specification, including performance and design characteristics and environmental conditions under which the software is to perform
- Interfaces external to the software module/ component/ service
- Testing requirements
- Privacy and security specifications
- Data definition and database requirements
- User acceptance and implementation requirements for the delivered application, including:
    - Will data validation from loads/migrations be required?
    - Test results required for quality acceptance
    - Test results required for performance acceptance
    - How will ease of use/efficiency be measured?

Operation and execution requirements:

- How often will the application be accessed?

- Does this application rely on the existence of another module/ data?
- Storage needs and potential growth
- What tools or technologies will users need to access the system (e.g., VPN, special licensing, etc.)?
- Maintenance requirements.

Walkthroughs and reviews:

- Architecture checkpoint review
- Internal peer reviews.

Design

Document the following:

- Identify tasks, pages, reports, procedures and functions
- Identify common modules that will be used
- Define the control logic for each task and unit
- Identify database and access requirements
- Identify feeds into the system module (including those coming from other sources)
- Identify exports and outputs from this system/module
- Assess performance requirements
- Define backup / recovery requirements
- Define the "hows" for each process rule and edit item identified in the analysis. These will become the logic descriptions in detailed design
- Define exception handling
- Conduct session(s) with architects (application and data architects) to review and flesh out design requirements
- Conduct final design walkthrough
- Document testing considerations: these requirements include identifying data files accessed, tests and startup considerations
- Define how the user will access the system/modules

Deliverables:

- Design document
- Project work schedule
- Coding specifications to be given to developers
- Transitional documentation for operations group (format TBD).

Coding/ Development

SQL and .Net are the standard development platforms for enterprise applications.

Stored procedures/functions are required for all Database access (server side). Client side coding is kept to a minimum and only used if approved by Application Architect and Business Systems Analyst. This is a security measure used to prevent infusion attacks.

All code must be written with the following goals in mind:

- Maintainability; adaptable to cope with changing requirements. The following questions will help judge the maintainability code:
  - Can I find the code that is related to a specific problem or change?
  - Can I understand the code? Can I explain the rationale behind it to someone else?
  - Is it easy to change the code? Is it easy for me to determine what I need to change?
  - Can I make a change with only a low risk of breaking existing features?
  - If I do break something, is it quick and easy to detect and diagnose the problem?
- Dependability. Does the code meet its specification, i.e., "correct output for each possible input"
- Efficiency. Is the program efficient enough for the environment in which it is used?
- Usability.

New stored procedures must include all the elements in the SQL template. Templates are located in the department share (see References and Related Documents for location).

Functions must be authorized and created by a DBA. Functions can be resource intensive depending on use so require a DBA to ensure efficiency and that their use is warranted.

Triggers must be authorized and created by a DBA. Triggers can be resource intensive depending on use so require a DBA to ensure efficiency and that their use is warranted.

User controls must be created/maintained by a Development Team Lead(s).

Comments/tracking: each code module must contain:

- Name of the module

- Purpose of the module
- Brief description of the module
- Original author
- Original implementation date
- Change control list which identifies:
    - Date of change
    - Who authored the change
    - Footprint ticket or project number that initiated the request for change
    - Description of change
    - Sample execute statement
    - In-line comment of variables at declaration to identify purpose and use
    - In-line comment blocks to describe required functionality for code when logic is complicated or more involved than usual. This adds to maintainability, allowing others to quickly understand what is happening in the logic.

Create / update unit tests documentation (at minimum) for modules/ features.

Create / update help documentation associated with the change to the module/ feature.

Create/update technical documentation associated with the change to the module/ feature.

Naming conventions must be adhered to.

Code portability is a goal. Hard-coding is to be avoided whenever possible (it is recognized that in some cases this is unavoidable). Should hard-coding be unavoidable, it must be identified and reviewed with the Business Analyst for approval. The approval must be documented as part of the project documentation.

All development must complete a code walkthrough prior to being promoted to QA/Production. Walkthroughs should include: BSA and a Team Lead. DBAs should be included for complex SQL or when processing times need to be reviewed.

AODA requirements

All new systems/pages/reports implemented must adhere to AODA standards. An AODA checklist exists (refer to the

References and Related Documents section above for the checklist location).

SQL & .NET coding standards

Templates exist for SQL procedures (insert, update, delete, select). These templates include code that must exist in all SQL procedures used for data retrieval/update. Application session information is required to identify who is retrieving/ updating/ deleting data. Procedures used to fill screen drop down lists do not require session information when the data retrieved is not sensitive. For example, a procedure which returns a list of valid academic sessions or valid course durations or valid subject codes would not need to have session information as this is public information.

For batch procedures special coding is used to identify a batch run.

Testing

Each software change requires testing. The degree of testing will vary depending on the size and complexity of the change.

Each test must be supported by documentation. For simple, low risk changes, this may take the form of screen shots captured and put into a Word document.

At minimum, there must be a document identifying what was tested and signed off by an analyst and the primary user indicating the change was successfully reviewed and tested prior to implementing to QA and production.

Testing for AODA compliance is required for pages/reports.

Consult/Include Operations staff in the test phase for transition training.

Document your test

Identify the type of test you are performing:

- Unit testing
- Regression testing
- Load testing
- User acceptance testing.

Identify the functionality/features being tested. Identify the functionality/ features NOT tested for clarity.

Identify who is required to participate in the testing their role in testing. Consider:

- Analyst
- Developer
- Client/user
- Other (e.g., DBA, ISSG, Client Services).

Identify what needs to be in place before testing can begin:

- Is there specific hardware that is required, or a specific configuration that needs to exist?
- Is there software required for this testing?
- Does data need to be loaded/set up?

Identify the Pass/Fail criteria for each item tested:

- Steps / schedule of activities: identify tasks and dependencies between the tasks.

Deliverables:

- Completed AODA checklist.
- Test logs: Show items tested with inputs and resulting outputs. This should be derived from the test cases that were identified and documented in the coding/development phase.
- Issues Log: Record issues encountered during testing and corrective action taken.
- Summary: brief summary of results, metrics and any observations during testing, including participant input.
- Approvals and sign-offs:
    - To validate the system/change has been tested thoroughly, sign-offs from the Business Systems Analyst, system owner and any other testers are required.

Document implementation plan

Clearly define communication plan:

- **Communication plan to user community of new functionality**
- **Communication plan internally to ITS groups required to support the product**. Identify and communicate what constitutes post-implementation support vs.

operational support to the project team and users of the system. **This includes but is not limited to**:

- Change Advisory Board (CAB) need to be notified of the impending implementation (when and what is impacted)
- Operations group for ongoing support of the product
- ISSG to ensure proper backup/recovery processes are in place when production is live
- Client services, so they are aware of new/altered functionality and potential impact to the Brock University community.

Clearly identify dates for implementation (i.e., if phases exists, each phase's implementation date).

Clearly identify tasks required to get the product implemented:

- Who is required to perform each task
- The order in which tasks must be completed, including all dependencies
- Identify risks:
  - Involved with the implementation
  - If the implementation is not performed
  - Contingency plans associate with the risks.

Identify all data imports required

- Are imports to be scripted or manually applied and by whom.

Identify system configuration required before the product can be used in a live environment. Consider:

- Security roles
- Secure document/file repositories
- Scheduled backups.

Training:

- What training is required for the users of the system?
- Who will do the training?

Validation:

- Who will validate system functionality once the product has been implemented?
- Who will validate data accuracy once the product has been implemented?
- Identify scripts or procedures to perform validations.

Define the length of the post-implementation phase:

- 4 weeks can be used as a rule of thumb on large projects (1 year in duration) or complex projects. Adjust accordingly for smaller projects. For shorter duration projects, 1 or 2 weeks should be sufficient
- The post-implementation phase is supposed to give the user time to use all features and functions while resources are available to support any unforeseen issues.

Go Live Decision

Ensure the project team and primary user contact/primary stakeholders are in agreement with the implementation plan.

Ensure the primary user contact/primary stakeholders agree with all that has been completed and tested.

Get agreement to move to production.

Deliverables:

- Implementation plan document
- Transitional documentation for operations group (format TBD).

Post-Implementation

The post-implementation time is typically referred to as a stabilization period. Resources are still assigned to the project, typically with a lower percentage of time allowance, and available to make adjustments if issues arise.

Any work done at this point is to ensure the functionality and features that were in scope as outlined in the charter, requirements, specification and design are working as expected.

Changes to functionality and features are not part of this process, they would require a new project or enhancement requests be initiated through regular channels.

Maintain a log of issues

Identify the priority level of each issue (to be done in consultation with the system user and project team):

- Level 1: assign to team member(s) to resolve as part of the post-implementation of the project. This level can

be further broken down to prioritize the sequence to work on if multiple issues exist with this priority level.
- Level 2: create Footprint tickets to be handled as part of the Operations team's regular mandate.

Ensure Operations team is kept aware of the issues log.

Prepare a project closing report identifying lessons learned and milestones.

Hold project close meeting and obtain signature on completion of report to close project.

Deliverables:
- Issues and resolution log.
- Project close report.

Templates

Templates for creating SQL procedures are to be used to create new routines. These templates are shells set up to include basic standards such as routine naming convention, comment block and application session coding.

Specifications / Requirements checklist

☐ Was thought given to the system administration functionality?
☐ Was thought given to error handling?
☐ Does the specification clearly divide the project into phases?
☐ Do all the phases have verifiable (and preferably undisputable) outcomes?
☐ Does the document refer to any related documents as specifically as possible? (Document title, revision, page number)?

If there are interfaces:
☐ Have the necessary data required for interfacing been identified?

☐ Is the maximum load (data and system usage) estimated?
☐ Are the security requirements specified?
☐ Are the operation and maintenance requirements specified (service transition)?
☐ Are the education/training requirements specified?

- ☐ Are the installation/migration requirements specified?
- ☐ Has there been a peer-to-peer review (walkthrough)?
- ☐ Has the application architect reviewed (walkthrough)?
- ☐ Have the requirements/specifications been agreed to and signed off by the user?
- ☐ Have reporting requirements been clearly identified?

Design checklist

- ☐ Is the design as simple as it can be?
- ☐ Are all the functions/features that are listed in the requirements covered?
- ☐ Are all assumptions, constraints, design decisions and dependencies documented?
- ☐ Have all reasonable alternative designs been considered, including not automating some processes in software?
- ☐ Does the design have features or functionality which were not specified by the requirements (e.g., are all parts of the design traceable back to requirements)?
- ☐ Does the design create reusable components if appropriate?
- ☐ Are modules well-defined including their functionality and interfaces to other modules?
- ☐ Interface details:
  - ☐ Routine name, parameters and their types, return type, pre- and post-condition, usage protocol with respect to other routines
  - ☐ File name, format and permissions

- ☐ Are all major data structures described and justified?
- ☐ Are major data structures hidden with access functions/procedures?
- ☐ Is the database organization and content specified?
- ☐ Are all key algorithms described and justified?
- ☐ Are all major objects described and justified?
- ☐ Is the user interface modularized so that changes in it won't affect the rest of the program?
- ☐ Is a strategy for handling user input described, i.e., file input, manually entered through filters, etc.
- ☐ Are key aspects of the user interface defined?
- ☐ Are space use estimates and a strategy for space management described and justified?
- ☐ Is a strategy for handling I/O described and justified?

- ☐ Is a coherent error-handling strategy included?
- ☐ Are error messages managed as a set to present a clean user interface?
- ☐ Are necessary buy vs. build decisions included?
- ☐ Is this designed to accommodate changes/enhancements in future?
- ☐ Is any part over- or under-designed?
  - ☐ Are the major system goals clearly stated?
  - ☐ Does the complete architecture fit together conceptually?
  - ☐ Is the top-level design independent of the machine and language that will be used to implement it?

- ☐ Are you, as a programmer who will implement the system, comfortable with the design?
- ☐ Design review and Walkthrough completed with architects (data and application)
- ☐ Instructions/documentation for transition to Operations team

Development / Coding checklist

- ☐ Does every input that comes from an untrusted source (i.e., typing into fields on a page, external systems) have associated error checking accounted for?
- ☐ Are all forms of validation done on the server side? (only allow on the client side on an as needed basis)
- ☐ Stored procedures used as the method for data validation/delivery
- ☐ Is each coding module of sufficient size? Limit the size for readability and maintainability. Use a 4 page rule of thumb. If the module is larger than 4 pages consider if it can be reduced in size or functionality can be split out; also consider the following:
  - Size and quantity of data that would need to be passed between routines
  - Number of temporary tables that would be needed
  - Any extra database reads/writes that would be required.

Does the code have the following:

- ☐ Proper naming convention
- ☐ Purpose is documented
- ☐ Brief description

- ☐ Original author and date are identified
- ☐ Change control area showing date of change, reason, person who did it and associated project or ticket number
- ☐ Sample execute
- ☐ Unit test documented and repeatable
- ☐ Sufficient commenting exists throughout the code to make it readable and understandable (i.e., maintainable) and the comments match the actual code.

Testing

- ☐ AODA requirements have been tested (see AODA checklist)
- ☐ Test case(s) have been identified
- ☐ Pass/Fail criteria has been identified for each test case
- ☐ Page/Report filters
  - ☐ Data entered in filters is trimmed
  - ☐ Each valid option is tested for each filter
  - ☐ Invalid data entered to test error control
- ☐ Test security by changing roles