## Philosophy

The foundation of our custom agile process was built from best of breed software development practices based upon firsthand experience and industry innovations. The key to a successful Agile project is a strong team of motivated engineers capable of collaborating effectively with stakeholders. Developers and stakeholders work together throughout the entire project to ensure that all requirements are properly satisfied and pre-defined goals are achieved. Software is delivered frequently for stakeholders to review and provide feedback to the development team for quick implementation of requested changes. Daily development operations are discussed openly in routine Scrums that keep engineers apprised of the project environment and task assignments. Our implementation of Agile is organized into tailored, well-defined iterations that are repeatable throughout the project lifecycle. Change happens and our approach embraces this indisputable fact.

**Implementation**

Scrum to manage the development process. Trevor??

Well defined development iterations, or sprints, are used to set goals (beginning), develop software (middle) and deliver a product (end). This process is repeated until project completion. Throughout every sprint we apply our tailored Agile Blueprint so that each sprint is consistent and standardized.

Iteration consists of??

At the beginning of an iteration a task list is created to define the goals and tasks that will be accomplished for the current iteration. Begin work

## Documentation

Software projects are affected by many external factors which cause changes to requirements, timeframes and technologies. Initial efforts to produce a large repository of documentation capturing requirement and design artifacts is typically a wasted effort that must be continually reworked as a project evolves. Our approach is to capture initial requirements and design artifacts during a short project kickoff period in order to begin development as quickly as possible.

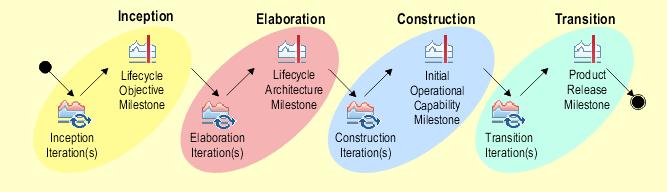
Clearsoft subscribes to an Agile Model-Driven Development (AMDD) approach. Modeling is a key visualization and design component to our engineering process. Visual models are created to represent all aspects of the system being built. They are also used as a communication aid between developers, architects and stakeholders so that technical and non-technical team members have a thorough understanding of development decisions. We create critical and useful documentation throughout every iteration of our Agile process. This approach results in a thorough portfolio of artifacts that represents complete system documentation at time of delivery.

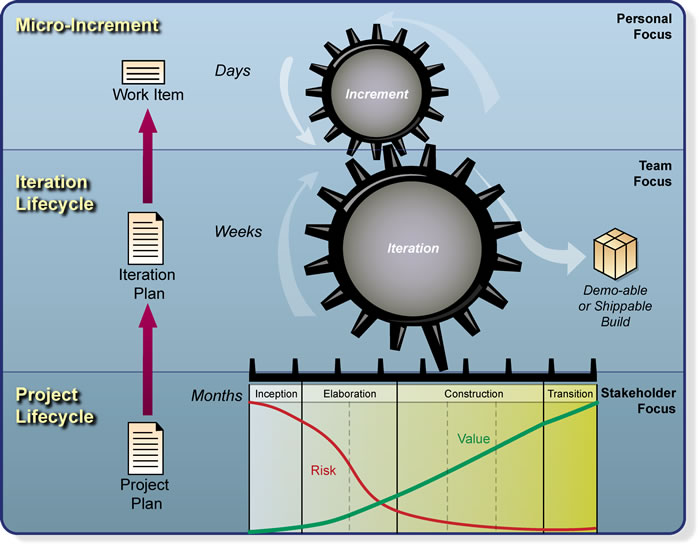
**Governance**

Agile iterations with consistent collaboration pull back the curtain on the development project for stakeholders. The result is better control over project scope, direction and scheduling for the stakeholder. Progress is measured by the delivery of working software after an iteration to allow stakeholders to make informed decisions and incremental changes.

Formal phases for the project are defined to be flexible but also crystalize an overarching picture of the project. Each phase encapsulates our iterative process.

* Inception phase involves initial requirements gathering and architecture strategy. An overarching vision is solidified and agreed upon between the stakeholders and technical team members.
* Elaboration phase takes the first steps of implementing the conceived architecture. A simplistic prototype is built to prove the validity of the architecture and provide the stakeholder a glimpse of how their requirements will be met.
* Construction phase
* Transition phse





scaleability

**Change Management**

Change happens during almost every phase of a software project for a variety of reasons. Bugs are discovered, new requirements are introduced, funding changes and changing priorities and just a few reasons. The cost of change is greatly reduced by a process that is nimble enough to respond quickly to new obstacles. This is a stark contrast to traditional waterfall approaches where change causes a chain reaction of very costly efforts to re-design, re-document and re-work all efforts prior to the introduction of the new change.

Each iteration of our Agile process includes all the steps needed to respond to any changes. Each change is prioritized and documented through the normal process at the beginning of a sprint. This approach also provides a greater degree of control to the stakeholder who must be an integral part of the Agile process by participating in every iteration.

## Blueprint

Our Agile software delivery blueprint is strategically tailored to each customer's environment. Every organization is unique and requires a flexible plan, or blueprint, that can be molded into a repeatable process which will yield consistently successful results. By incorporating components from a toolbox of industry proven techniques, patterns and technologies we are able to build effective agile processes from our blueprint.

* Requirements - Business requirements are captured and documented. Engineers meet directly with stakeholders to discuss and white board requirements. A variety of documentation and models, like Use Cases, are produced during this phase.
  + Stakeholders are engaged during all iterations to keep the project on track.
* Design – Architectural modeling artifacts are built as needed for the development team.
  + Technical planning results in clarity for the development team.
* Development and Continuous Integration (CI) – Quality control measures are applied through automated process continuously throughout all iterations.
  + Defects are reduced throughout the iterations instead of during one massive testing phase.
* Automated software builds - (tool: ANT). Software is compiled and built with automated tooling, like ANT.
  + Benefit:
* Automated software deployments to multiple environments. Using tools like CruiseControl we build an infrastructure that automates the deployment of software to enterprise servers like Websphere or Weblogic.
  + Deployment errors are eliminated and support costs from server support teams are reduced because automation is reliable, effective and repeatable.
* Automated white box testing to prove code functionality (tool: JUnit).
  + Software bugs are identified early resulting in higher quality code delivered to independent testers. Costs are reduced because quality code requires fewer testing cycles.
* Automate code verification using custom rules to detect bugs and enforce standards (tool: PMD).

**References and sources of inspiration for the Clearsoft process:**

Agile Manifesto (Where it started) - <http://www.agilemanifesto.org/>

Agile Industry (Best practices) - <http://www.agilealliance.org/>

Agile Principles (Be Agile) - <http://martinfowler.com/>

OpenUP (Structure Agile) - <http://epf.eclipse.org/wikis/openup/>

Scrum (Manage Agile) - <http://www.scrumalliance.org/>

Agile Delivery (Deliver Agile) - <http://www.disciplinedagiledelivery.com/>

Agile Modeling (Visualize your project) - <http://www.agilemodeling.com/>

Stuff to include:

test driven

selc mapping

iterative

pattern based

flexible

repeatable

modeling

open source

cost effective

proving ground

inspiration - agile manifesto, ambler, fowler, DAD.

active stakeholder participation

best practices

scrum = agile in action (implementation of agile)

burn down

backlog