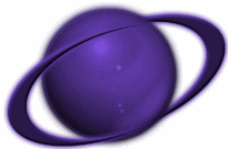


Object-Oriented Methodology 101

Presented by:

**William F. Nazzaro
Nazzaro & Associates**

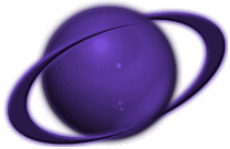
**E-Mail - bill@williamnazzaro.com
Phone - (610) 831-1151**



Object-Oriented Methodology 101

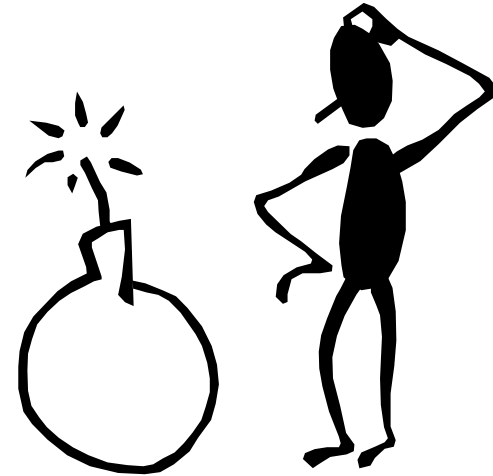
- Premises
 - *OO is becoming critical as systems become more*
 - » *Complicated*
 - » *Complex*
 - » *Distributed*
 - *Organizations are*
 - » *No longer investigating OO as a possibility*
 - » *Investigating how to use OO competitively*
- Objectives
 - *Discuss reasons/benefits of OO*
 - *Discuss the essential linchpin to realize those benefits*

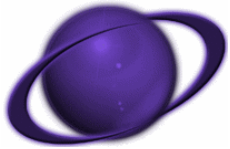




What to Expect

- Object-orientation's first impact at your company
 - *Can be a tremendous success, or*
 - *A complete blunder*
- Need awareness of this technology's
 - *Power*
 - *Impact*
 - *Shortcomings*
 - *Potential*



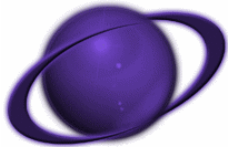


Benefits of Object-Orientation

- Faster development
- Reduced cost
 - *Reuse*
- Higher quality systems
- Improved architecture
 - *More adaptable*
 - *More scalable*



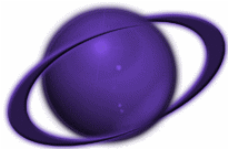
***Benefits come from careful analysis, design,
and development practices that exploit the OO paradigm!***



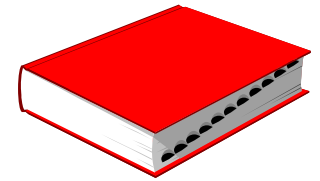
Methodology

The Quintessential Linchpin!!!

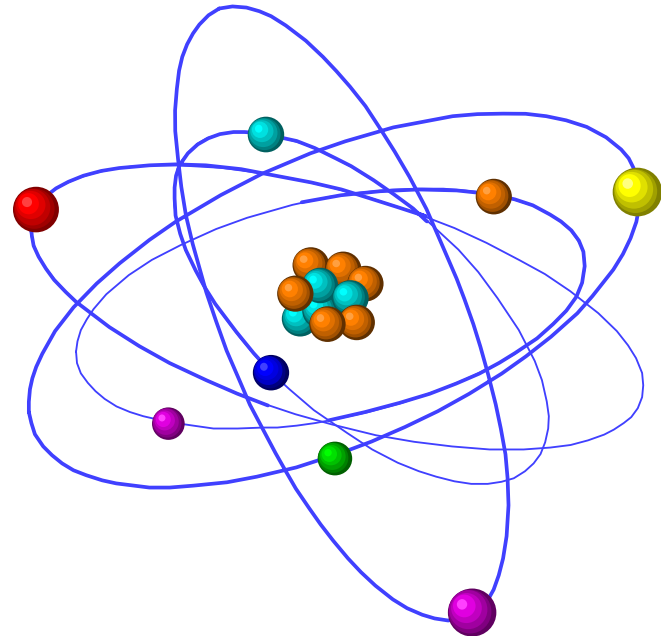




Methodology

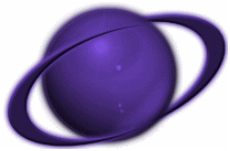


- A specification of a process through which software can be developed
- A set of generic software development techniques together with a set of guidelines defining how and when those techniques should be applied



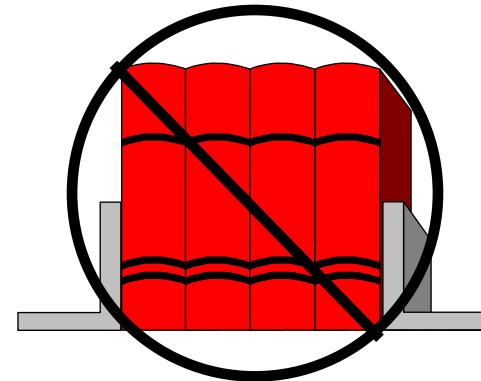
“Though there is madness, there is a method in’t.”

- Hamlet, William Shakespeare

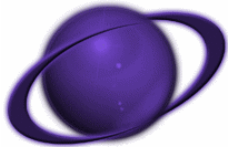


What Should an OO Methodology Do?

- ☑ Define
 - *Project roles*
 - *Software lifecycle*
 - *Deliverables to be created*
 - *Standards for documenting those deliverables*
 - *Techniques for creating those deliverables*
 - *Dependencies among deliverables*
- ☑ Ensure consistent application of techniques
- ☑ Facilitate staff transitions between phases and projects



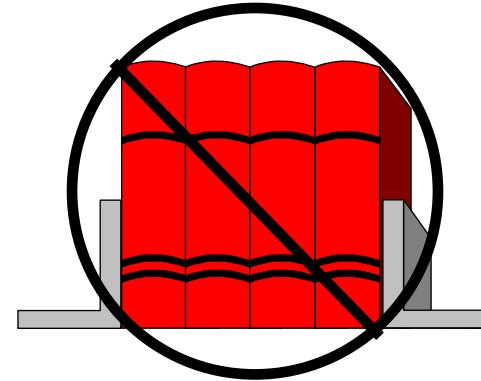
*Needs to be reasonable, pragmatic,
and most of all flexible*



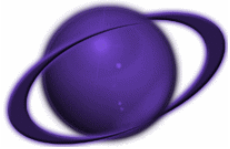
What Should an OO Methodology Do?

- ☑ Provide
 - *An experience base in the following areas*
 - » *Estimating guidelines and templates*
 - » *Techniques for analyzing models*
 - *Depth of coverage for life cycle phases*
 - *Guidance for*
 - » *Object-Oriented Applications*
 - » *Distributed Object Applications*
 - *Quality examples*

- ☑ Stress iterative and incremental development

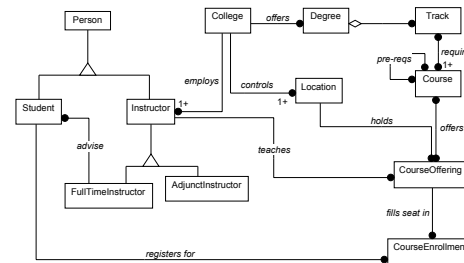
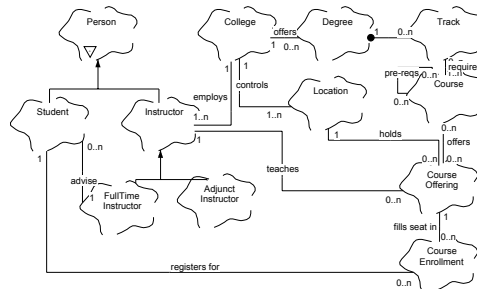
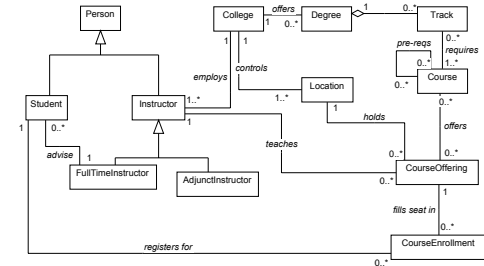


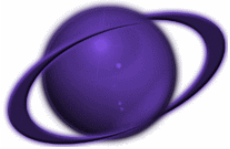
*Needs to be reasonable, pragmatic,
and most of all flexible*



Methodology Misconceptions

- Notation is not methodology
 - *Unified Modeling Language (UML)*
 - *Object Modeling Technique (OMT)*
 - *Booch*





Methodology Misconceptions

- Language is not methodology

- *C++*
- *Smalltalk*
- *Java*
- *Eiffel*
- *Visual Basic*

- Tools are not methodology

- *Rational Rose*
- *Paradigm Plus*
- *Select OMT*
- *GUI Builder*

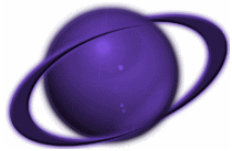
```
// FullTimeInstructor
#include "FltTmnst.h"

///  
begin module.additionalDeclarations preserve=yes
///  
end module.additionalDeclarations

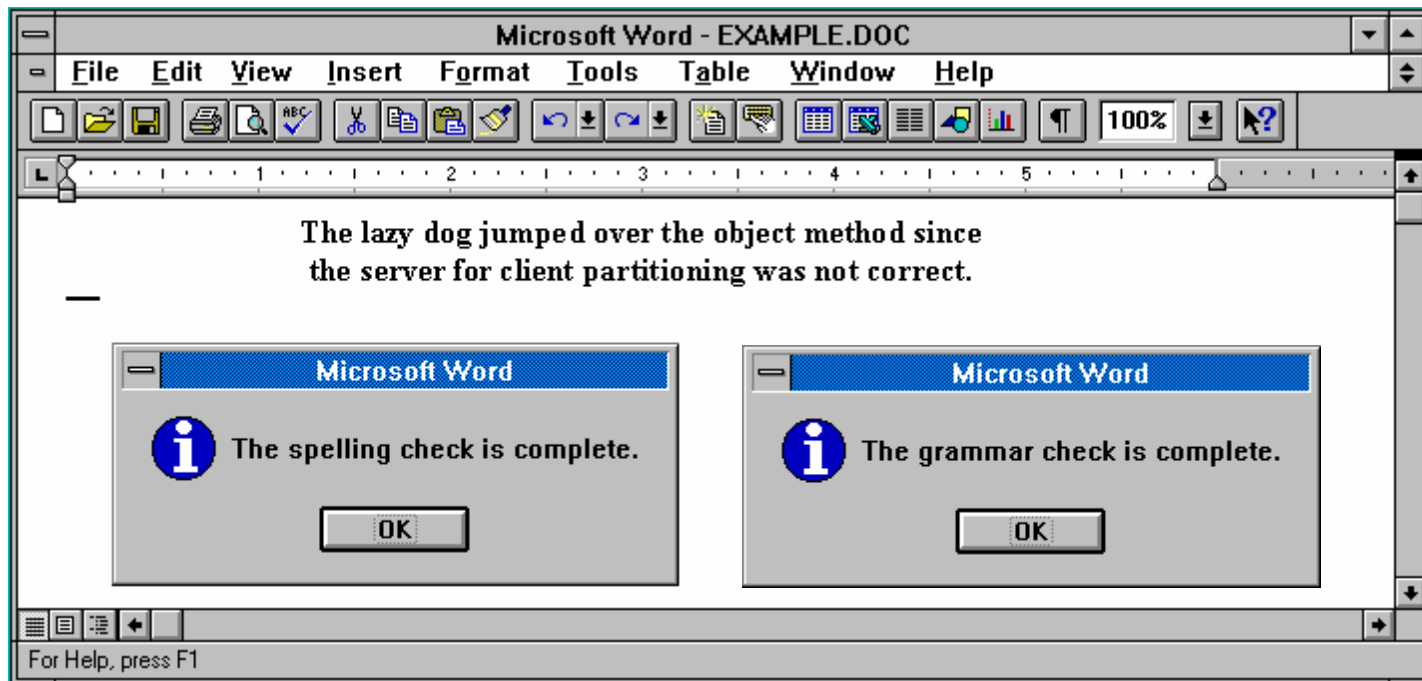
// Class FullTimeInstructor

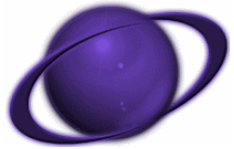
FullTimeInstructor::FullTimeInstructor()
  ///  
begin FullTimeInstructor::FullTimeInstructor%.hasinit
  ///  
end FullTimeInstructor::FullTimeInstructor%.hasinit
  ///  
begin FullTimeInstructor::FullTimeInstructor%.initialization
  ///  
end FullTimeInstructor::FullTimeInstructor%.initialization
{
  ///  
begin FullTimeInstructor::FullTimeInstructor%.body preserve=yes
  ///  
end FullTimeInstructor::FullTimeInstructor%.body
}

FullTimeInstructor::FullTimeInstructor(const FullTimeInstructor &right)
  ///  
begin FullTimeInstructor::FullTimeInstructor%copy.hasinit
  ///  
end FullTimeInstructor::FullTimeInstructor%copy.hasinit
  ///  
begin FullTimeInstructor::FullTimeInstructor%copy.initialization
  ///  
end FullTimeInstructor::FullTimeInstructor%copy.initialization
{
  ///  
begin FullTimeInstructor::FullTimeInstructor%copy.body preserve=yes
  ///  
end FullTimeInstructor::FullTimeInstructor%copy.body
}
```



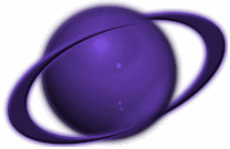
Tools Are Not Smart *So We Have to Be*





What to Look For in a Methodology

Process



Process*

Predictability vs. Creativity

- Management needs predictability,
 - *Essential for*
 - » *Team organization and assignments*
 - » *Incremental build plans*
 - » *Risk mitigation*
 - » *Proactive management*
- Development team thrives on creativity
 - *Essential for creation of solutions*
 - » *Innovative*
 - » *Unique*

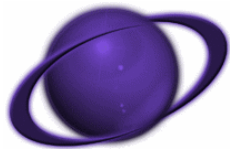
Problem - Need to establish a balance between these two extremes

Answer - Distinguish the macro/micro elements of the development process

↪ *Macro Development Process*

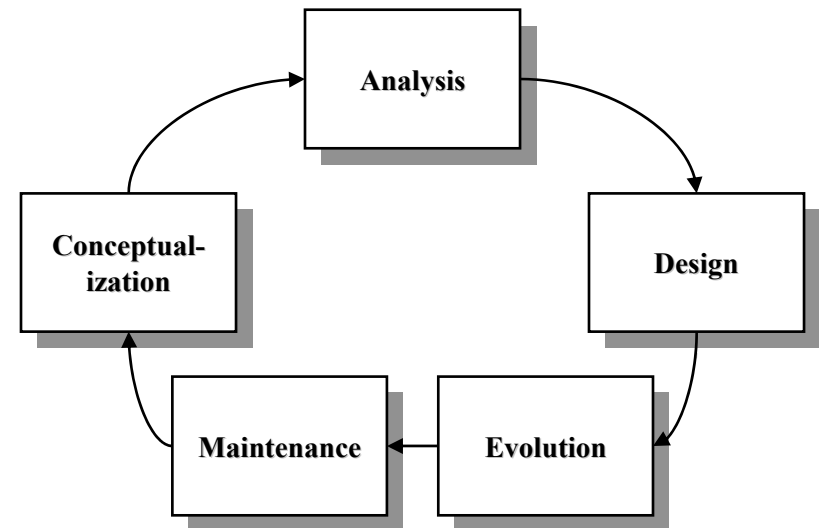
↪ *Micro Development Process*

* Adapted from: Grady, B., Object Solutions.

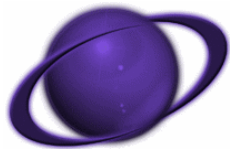


Macro Development Process *Overview*

- Overall concern is planning
- Three key elements
 - ① *Continuous integration*
 - ② *Executable releases that grow in functionality*
 - ③ *Releases enable management to measure progress and actively identify risks and mitigate them*
- Macro process
 - *Represents the project manager's needs*
 - *Controlling framework for the micro process*
 - *Time frame measured in weeks/months*



Project managers can only control things they can see



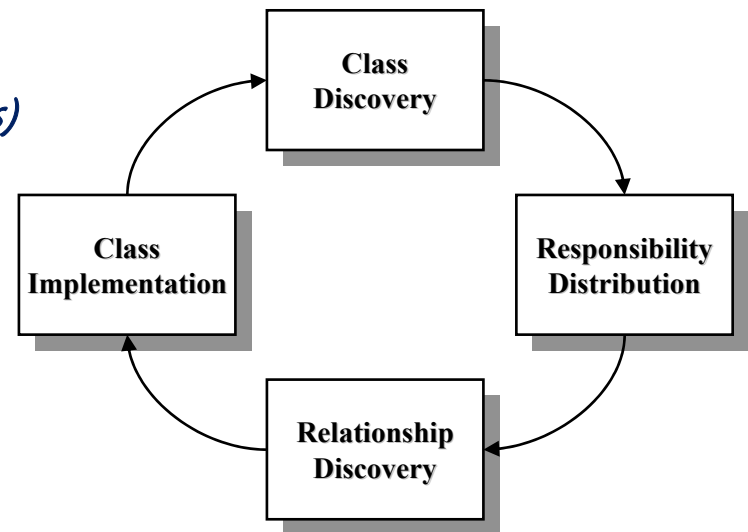
Micro Development Process *Overview*

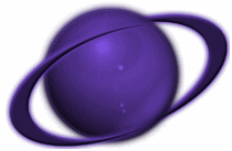
- Agenda

- ① *Select the correct abstractions*
- ② *Determine proper distribution of responsibility*
- ③ *Create a set of mechanisms that regulate these abstractions (e.g., semantic dependencies)*
- ④ *Concretely represent these abstractions and mechanisms*

- Micro process

- *Represents the development team's needs*
- *Carries on throughout the macro process, but each iteration has a different emphasis depending upon the project's current macro process*
- *Time frame measured in weeks/days*

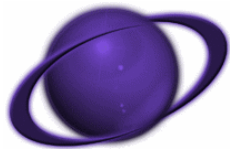




What Can I Expect on My First OO Project?

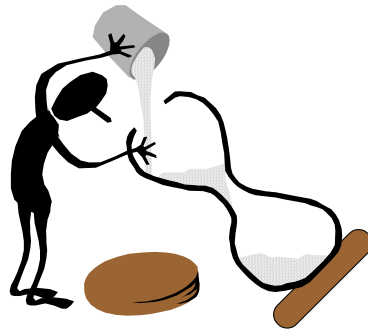
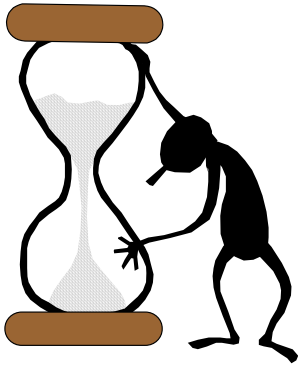
- Lessons Learned
 - *Project management*
 - » *Duration*
 - » *Staffing*
 - » *Reuse strategy*
 - » *Project estimating*
 - *Analysis and design*
 - *Technology*
- How does and OO Methodology Help?





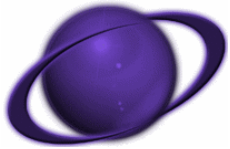
Lessons Learned

Project Management - Duration



- First project
 - *Will take longer than using your traditional approach*
 - *Mostly learning curve*
 - *Little or no reuse*
- Second project
 - *Will take longer than using your traditional approach*
 - *Learning curve still exists for design and architecture*
 - *Maybe some reuse*
- Third project (payback)

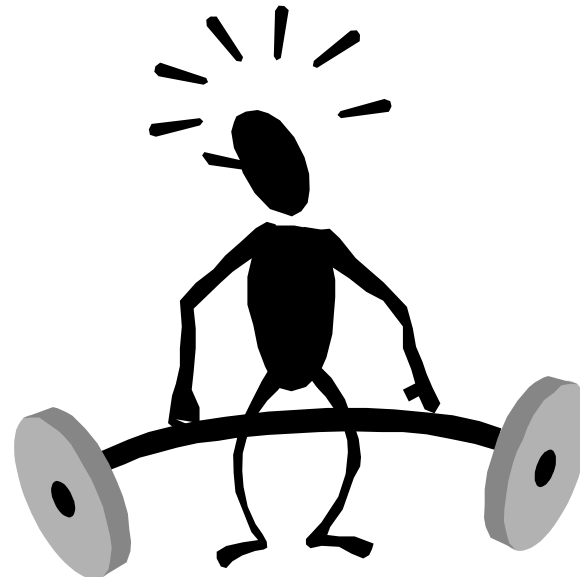
*Duration of a project is approximately 6 months

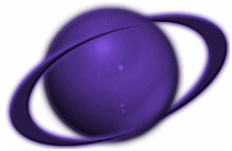


Lessons Learned

Project Management - Staffing

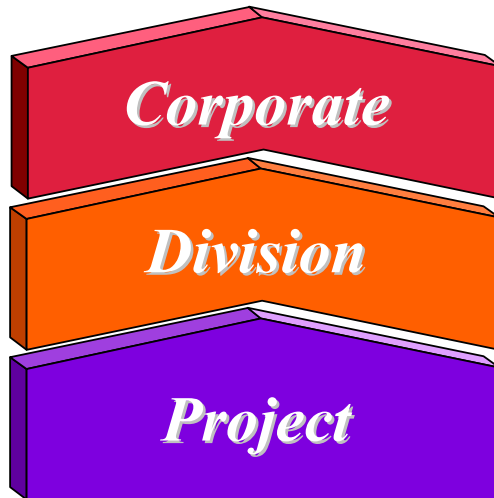
- Few skilled and experienced staff
 - *Options*
 - » *Hire an experienced staff*
 - » *Subcontract*
 - » *Mentor*
- Strip mining vs. investment
 - *“On the job learning” doesn’t work*
 - *Grow your experts*
- Realize everyone will not be capable of making the paradigm shift to OO



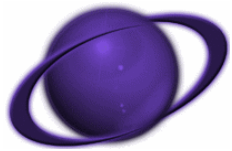


Lessons Learned

Project Management - Reuse Strategy



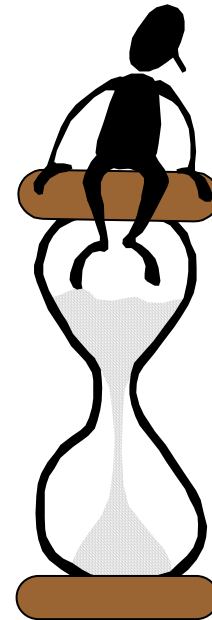
- Incorrect belief that reuse is guaranteed because we are object-oriented
 - *Must plan for reuse*
 - *Takes time and money*
- Need
 - *Specific procedures*
 - *Dedicated resources*
 - *Investment in re-use*
- Diligently select candidate classes to make reusable
 - *Every class does not have to be reusable*
 - *More reuse is not always better*
 - *Examine cost/benefit trade-offs*

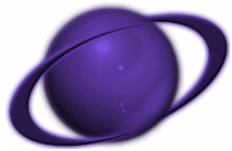


Lessons Learned

Project Management - Estimating

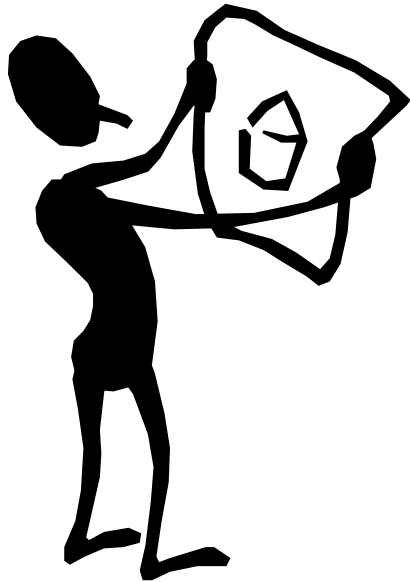
- Original estimates were based upon “Classic” development approach
- Few object technology projects as models
 - *Had to learn by experience*
 - *Rework necessary but valuable*
- Project plans need to include time for
 - *Prototypes and evaluations*
 - *Walk-throughs and inspections*



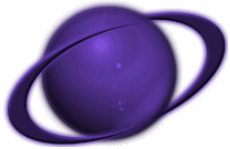


Lessons Learned

Analysis and Design - 1

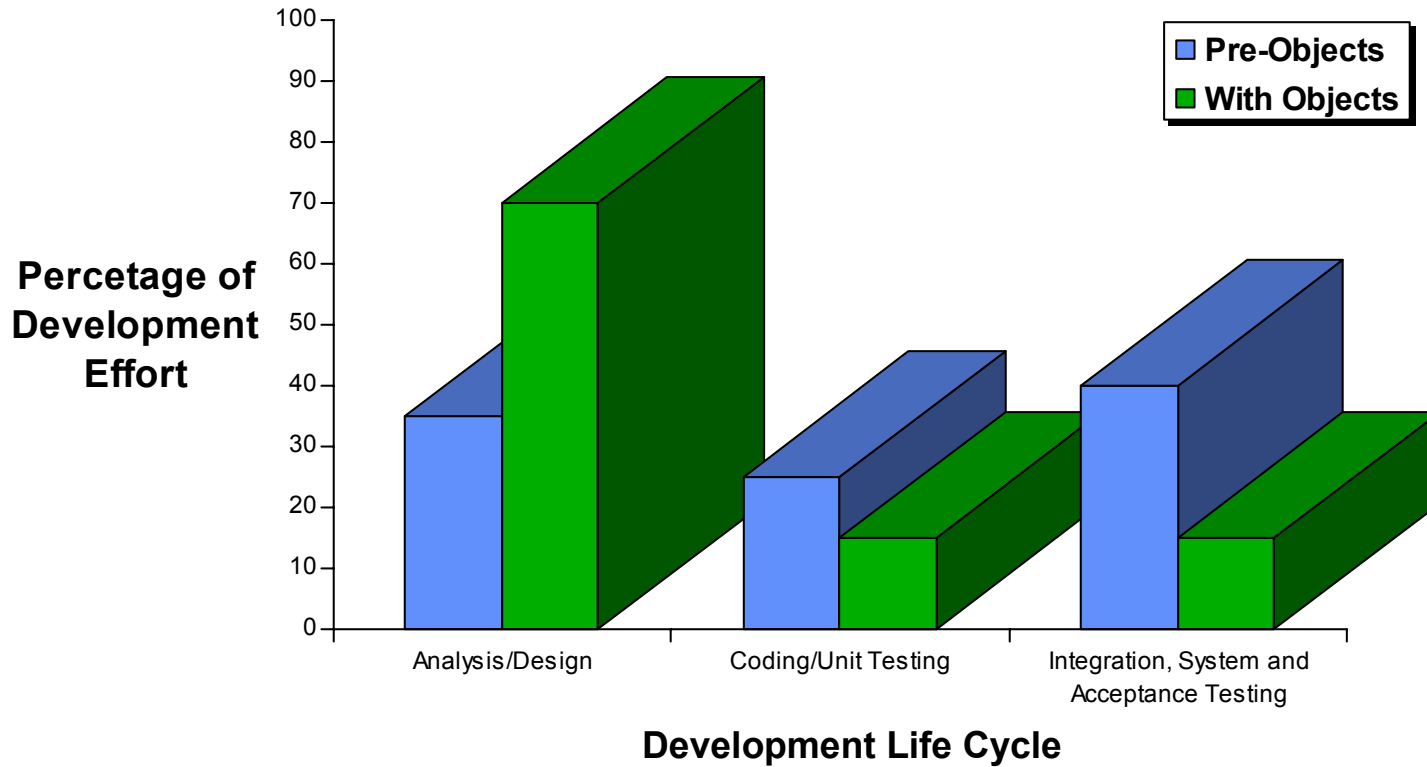


- Be aware of creating “object-free” applications
 - *Seduction by the “dark side” of OO*
- Empower a chief architect or architecture team
- Illusion of rapid progress during development
 - *Essentially the “Universal Studio® Effect”*
- Do not underestimate analysis and design
 - *Need*
 - *Difficulty*

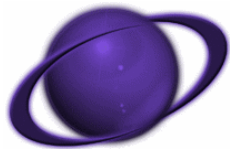


Lessons Learned

*Analysis and Design - 2**



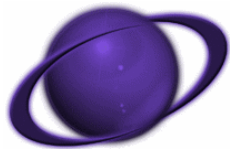
* Source: Gartner Group



Lessons Learned

Technology - Maturity?

- Individual technologies have matured but new technologies continue to develop and gain mindshare
 - *For example,*
 - » *Internet/intranet applications*
 - » *Object Request Brokers (ORBs)*
 - » *Object-oriented databases*
- Major risk - vendors failed to deliver products on time
- Software Development Environment
 - *Must be managed*
 - *Upgrades frequent*
 - *Upgrades not 100% compatible with earlier versions*

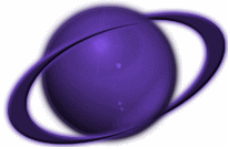


In Summary

Five Guidelines for Success

- ❶ Organization must understand their problem
- ❷ Upper management commitment
 - *Stand by the project*
 - *Accurate expectations*
- ❸ Select/implement a methodology
- ❹ Effective education/mentoring
 - *Multiple levels of education*
 - *Front-end lifecycle education*
 - » *Requirements*
 - » *Analysis*
 - » *Design*
 - *Language education*
 - *Testing*
- ❺ Select a pilot project and set accurate expectations

***Must be able to answer the question -
“Why are we transitioning to OO?”***



Object-Oriented Methodology 101

Presented by:

**William F. Nazzaro
Nazzaro & Associates**

**E-Mail - bill@williamnazzaro.com
Phone - (610) 831-1151**