SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

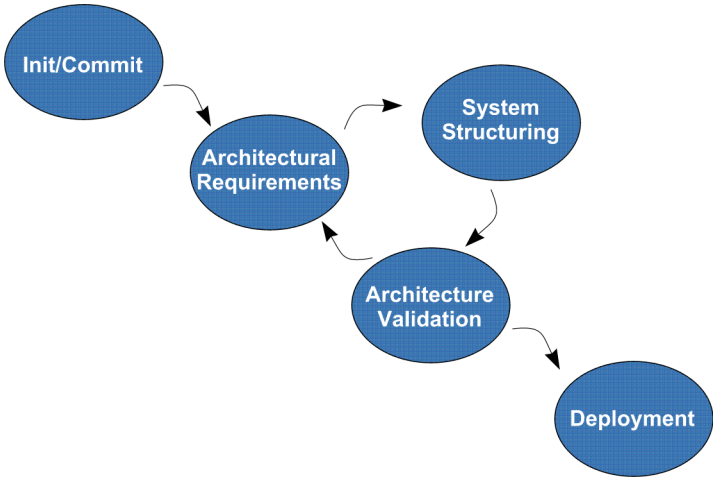
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

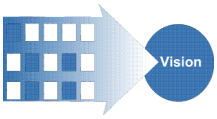
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

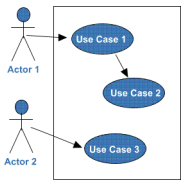
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

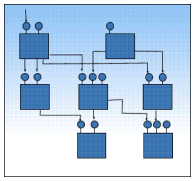
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

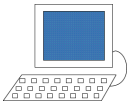
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

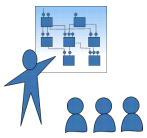
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

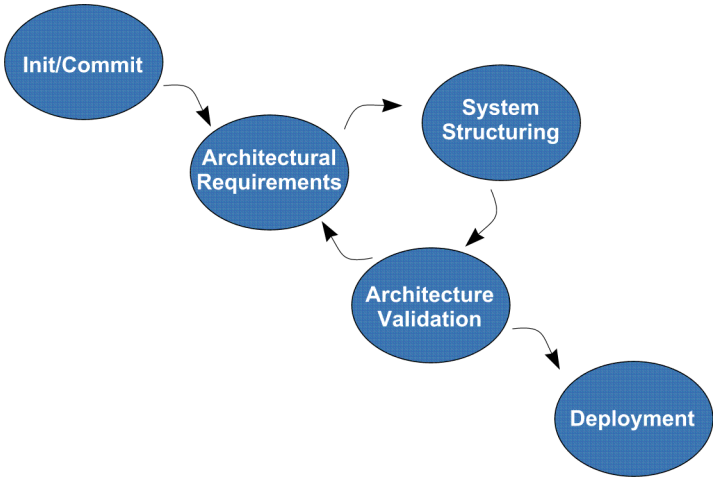
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

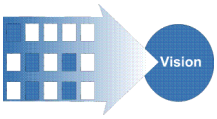
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

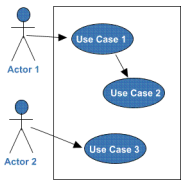
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

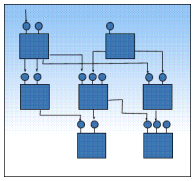
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

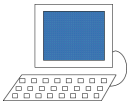
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

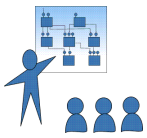
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

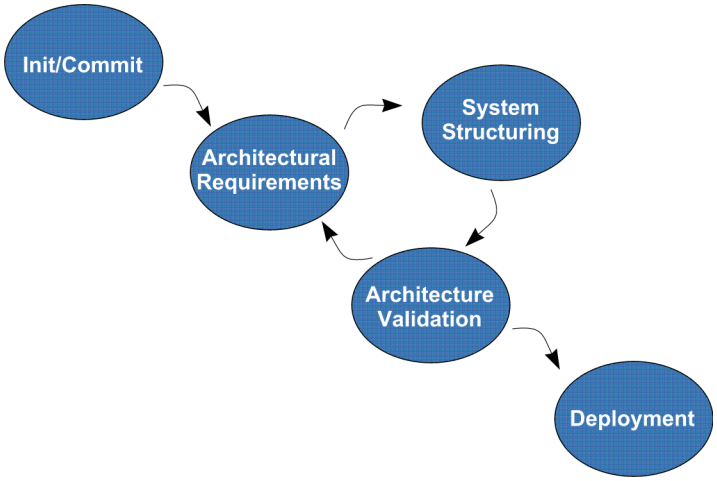
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

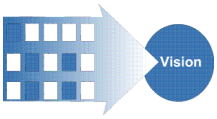
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

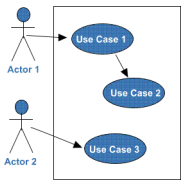
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

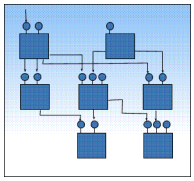
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

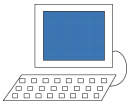
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

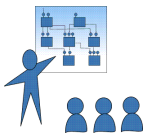
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

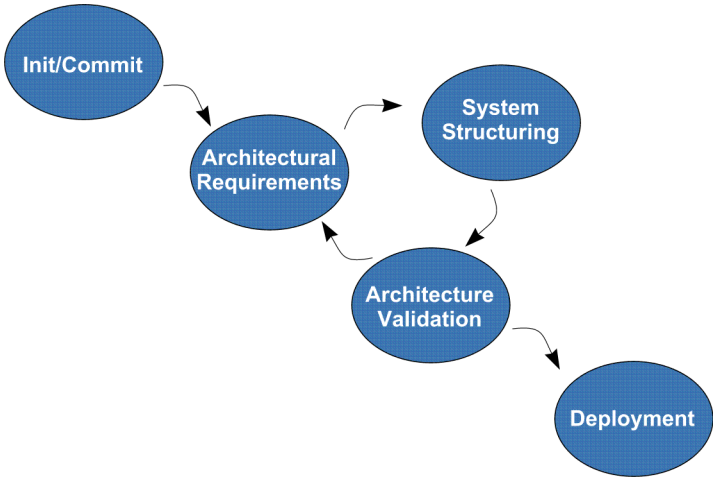
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

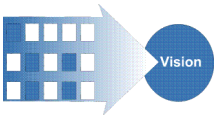
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

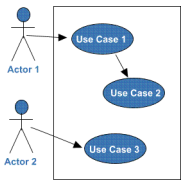
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

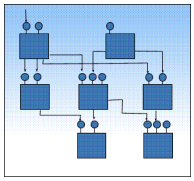
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

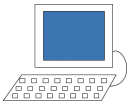
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

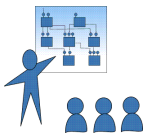
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

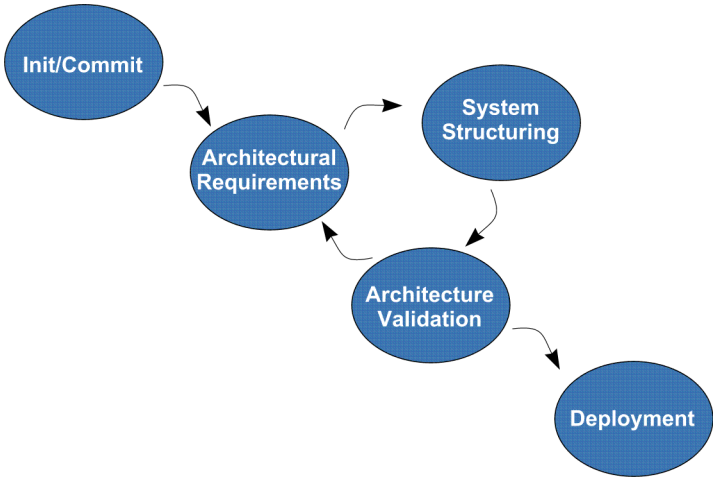
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

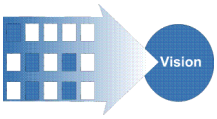
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

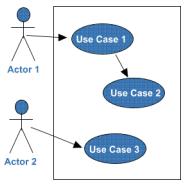
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

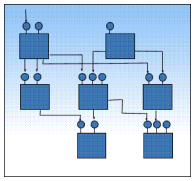
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

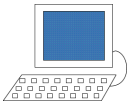
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

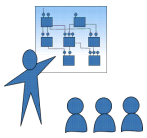
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

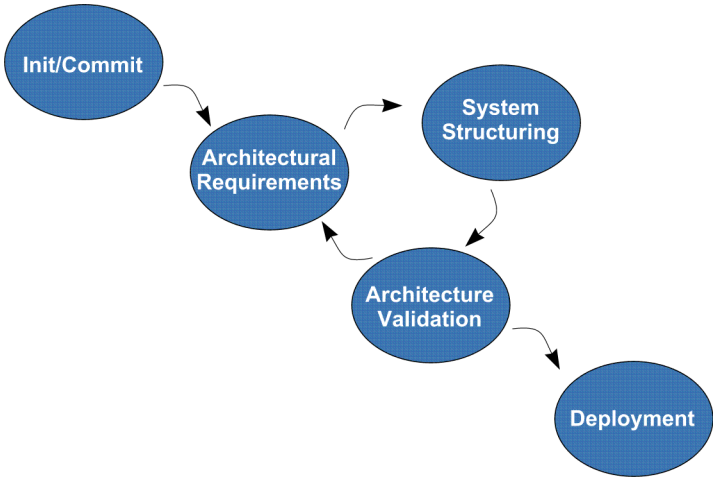
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

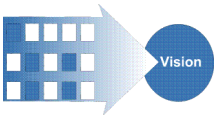
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

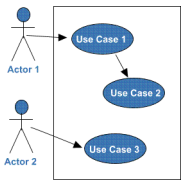
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

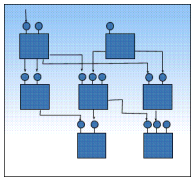
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

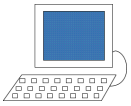
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

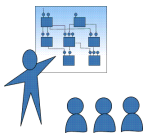
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

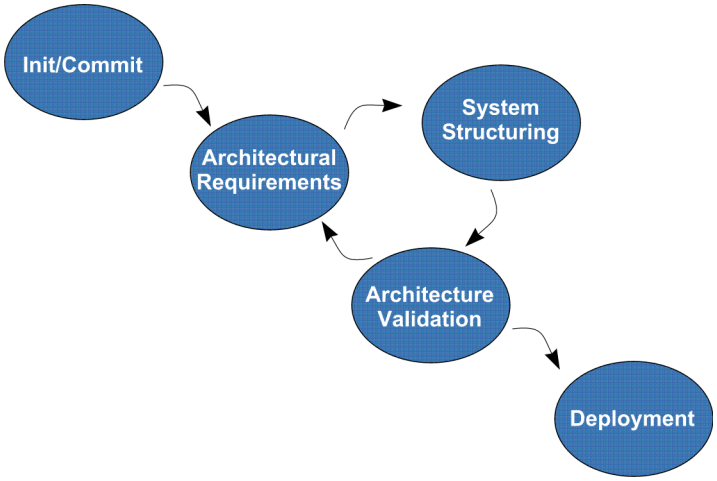
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

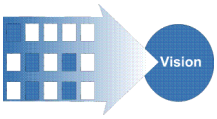
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

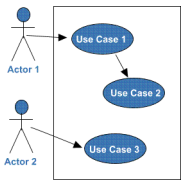
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

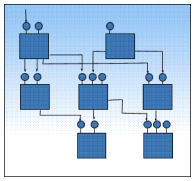
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

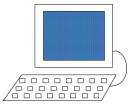
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

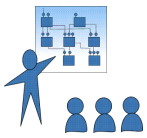
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

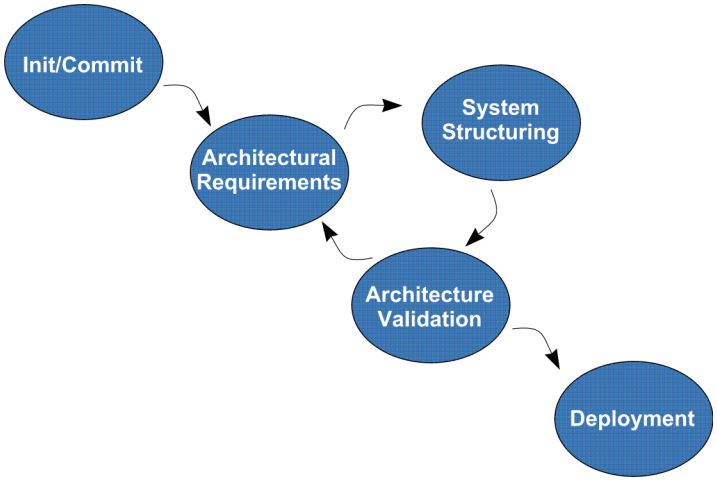
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

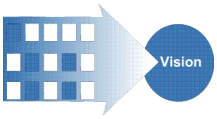
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

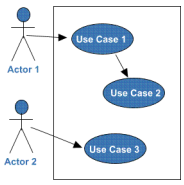
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

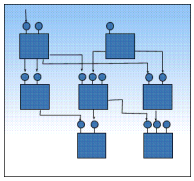
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

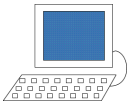
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

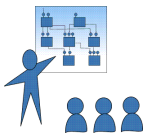
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

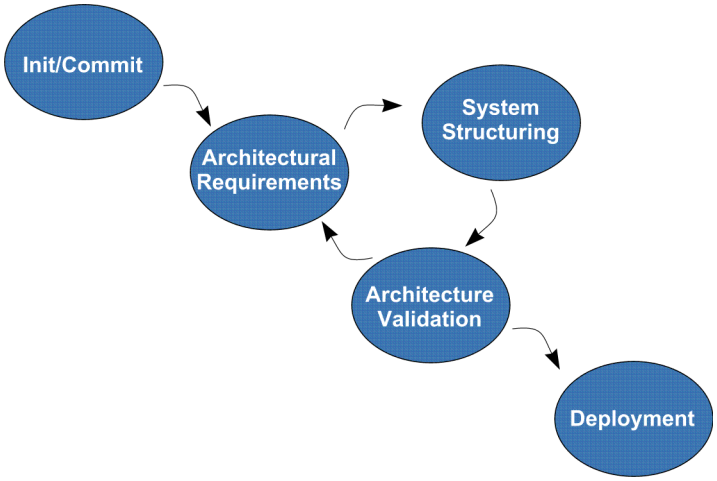
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

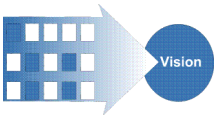
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

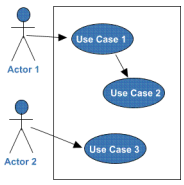
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

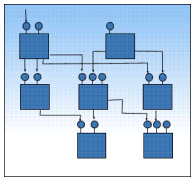
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

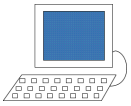
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

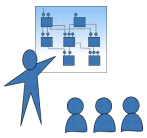
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

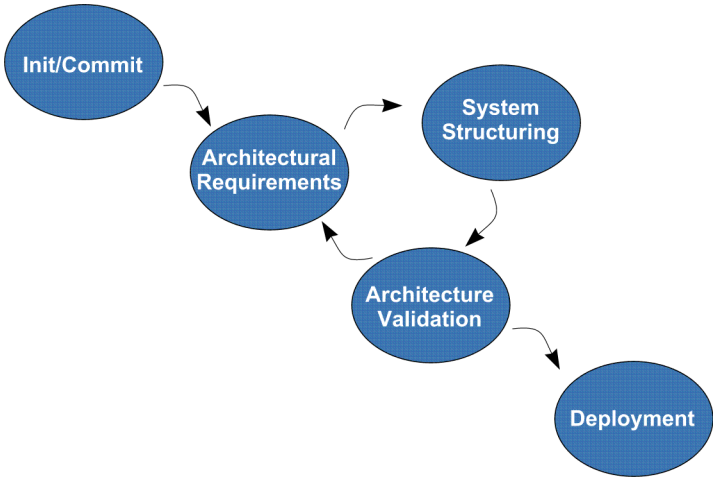
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

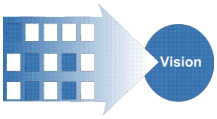
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

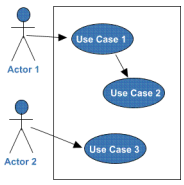
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

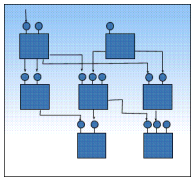
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

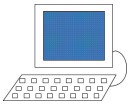
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

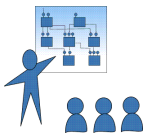
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

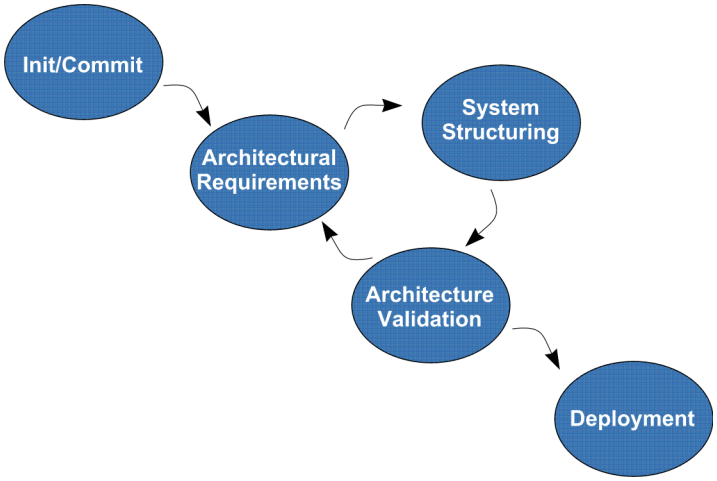
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

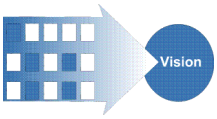
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

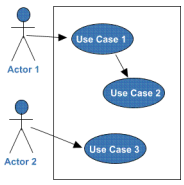
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

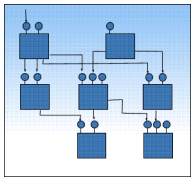
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

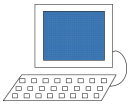
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

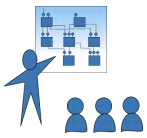
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

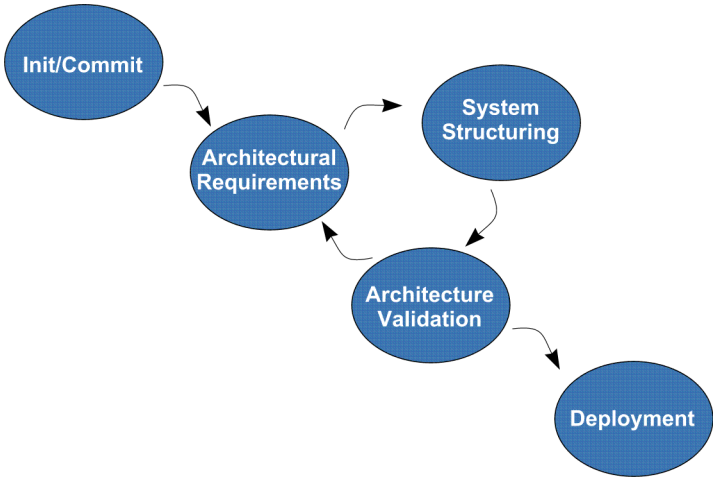
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

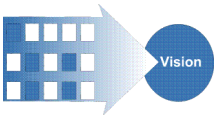
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

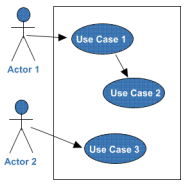
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

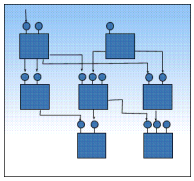
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

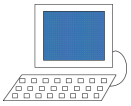
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

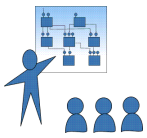
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

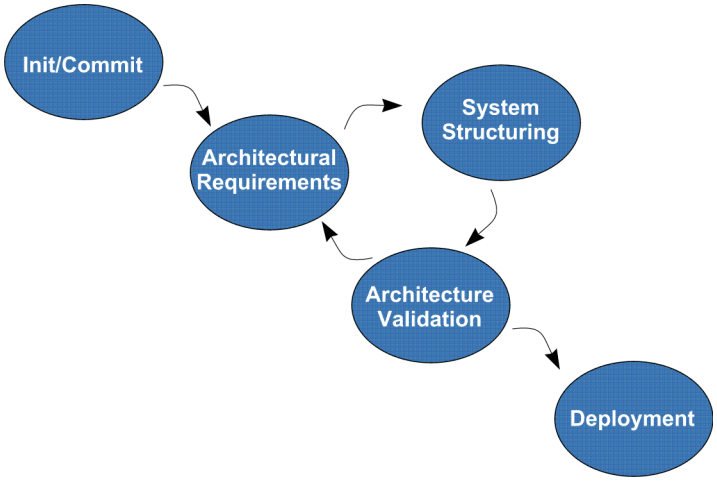
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

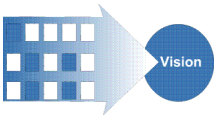
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

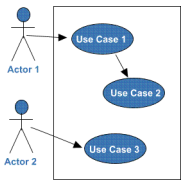
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

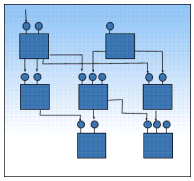
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

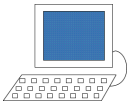
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

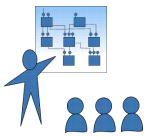
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

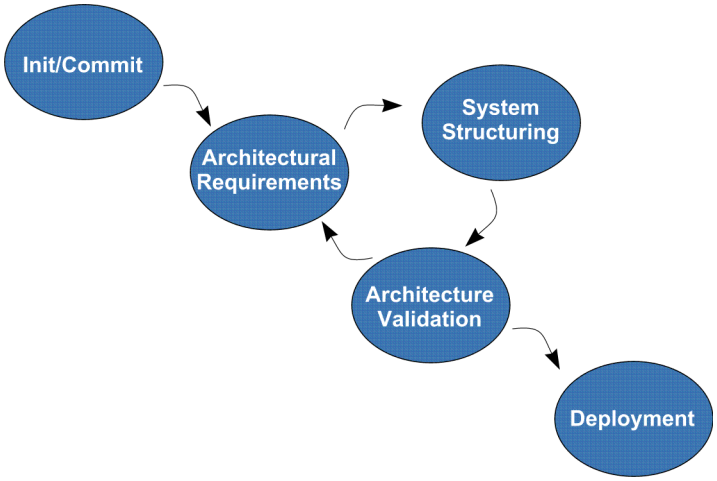
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

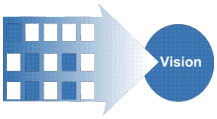
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

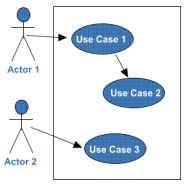
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

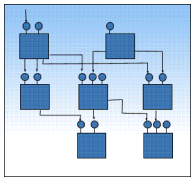
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

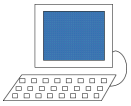
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

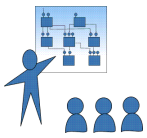
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

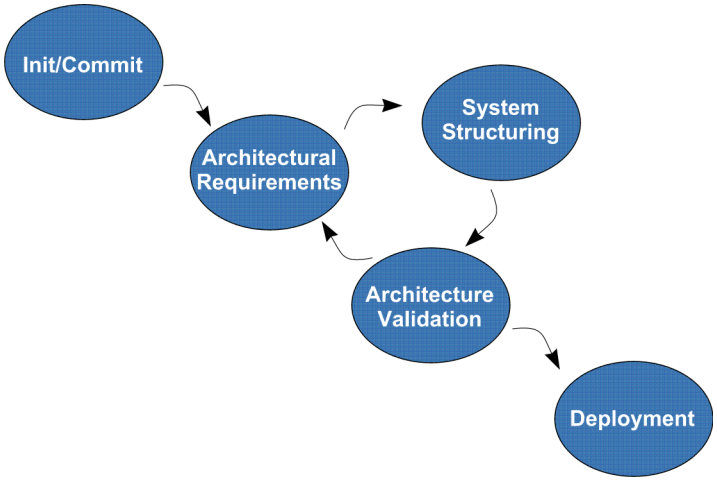
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

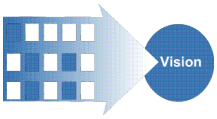
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

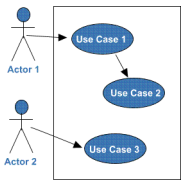
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

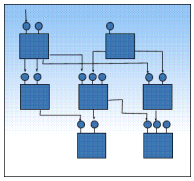
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

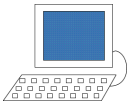
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

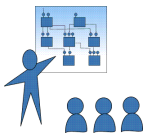
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

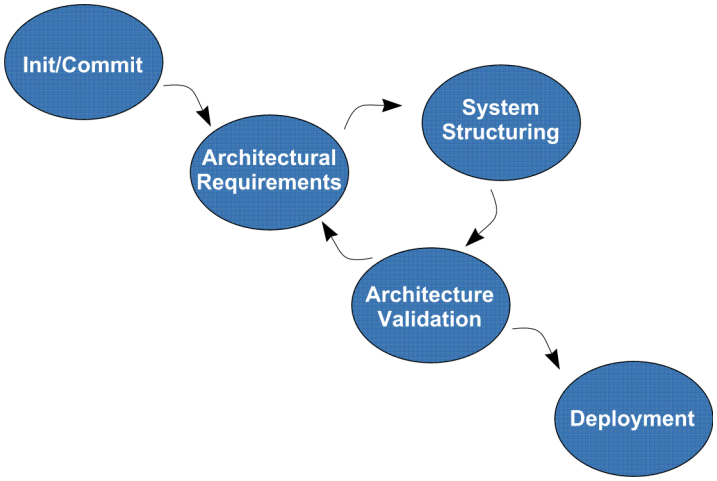
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

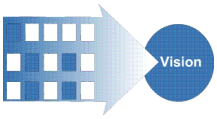
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

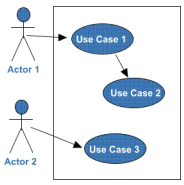
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

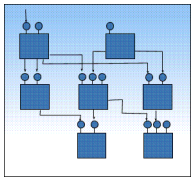
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

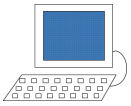
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

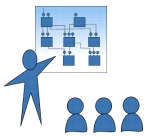
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

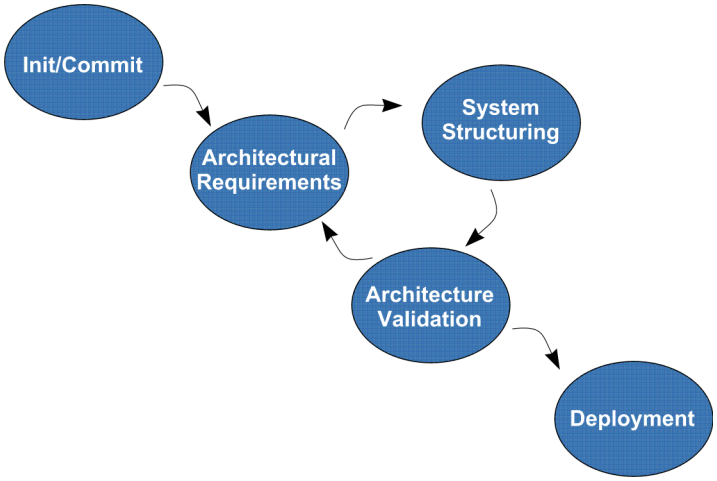
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

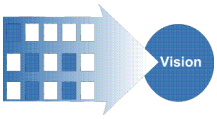
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

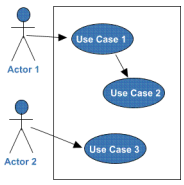
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

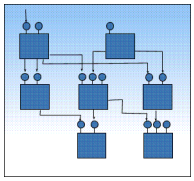
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

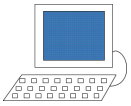
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

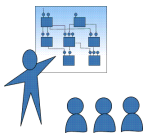
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

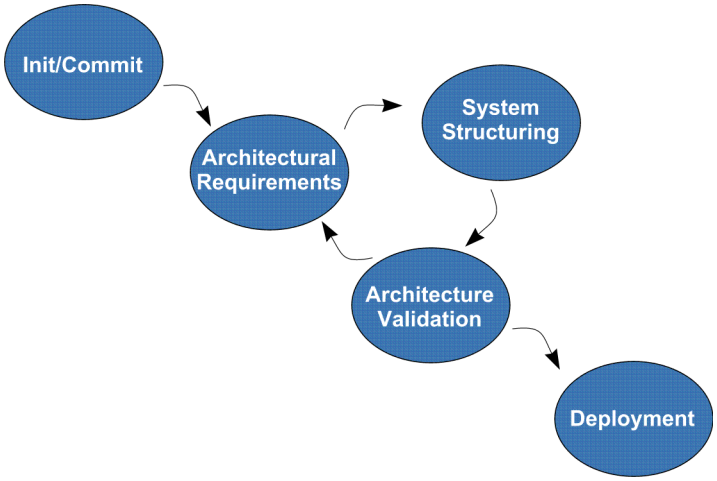
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

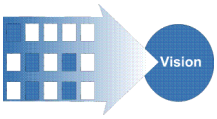
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

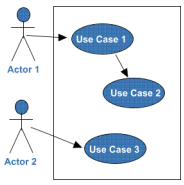
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

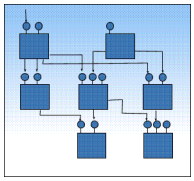
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

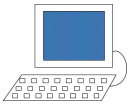
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

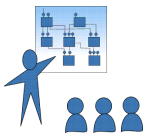
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

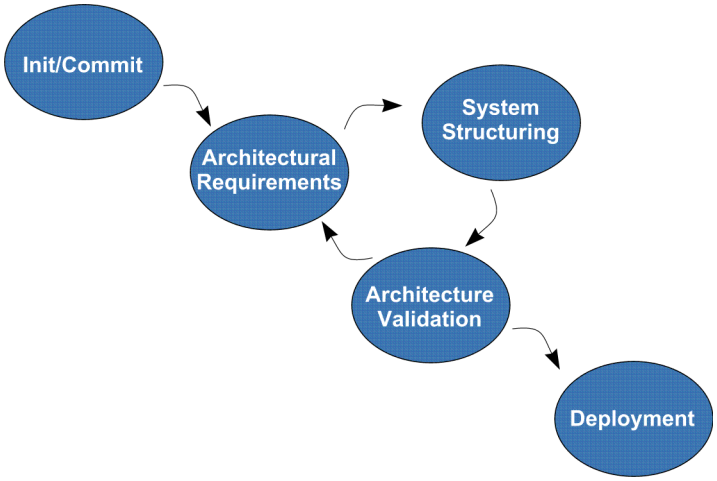
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

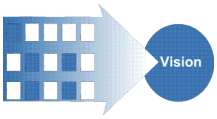
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

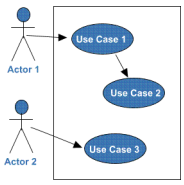
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

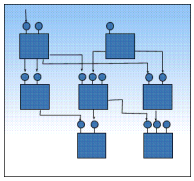
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

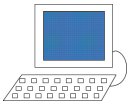
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

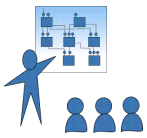
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com

SOFTWARE ARCHITECTURE

Architecture Architecting Architects

Action Guide

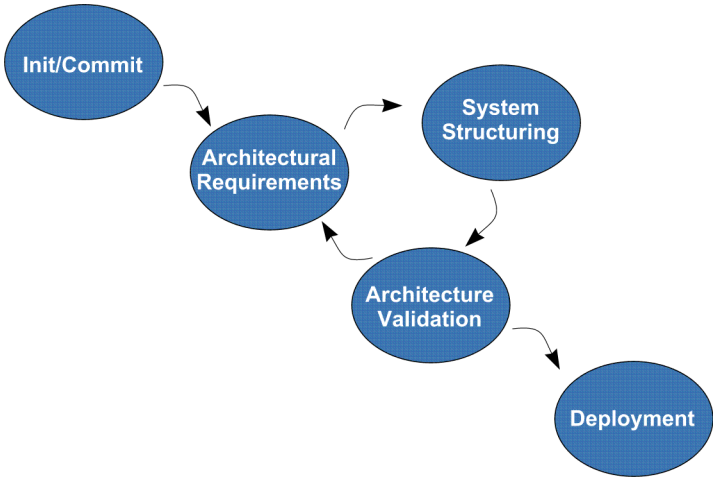
from Bredemeyer Consulting

Architecture

Software architecture is the high-level structure of a software system, comprising software components and the relationships among them. A good architectural description includes various views of the architecture, principles directing component design and evolution, and documentation of assumptions and rationale behind architectural choices linked to functional requirements and system qualities.

Architecting

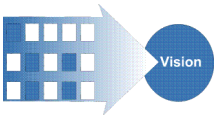
The software architecting process involves the following steps:



1. **Init/Commit**: Gain management sponsorship and form the architecture team
2. **Requirements**: Establish and document the architectural requirements
3. **Structuring**: Define the architecture
4. **Validation**: Validate that the architecture meets the requirements
5. **Deployment**: Deploy the architecture to the developer community

The architectural requirements, system structuring, and architecture validation steps are best conducted iteratively, as shown in the diagram. The process steps are described below. Also, see our ***Resources for Software Architects*** web site (http://www.bredemeyer.com) for more.

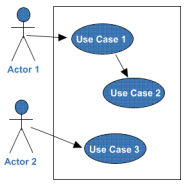
Init/Commit



Gain management sponsorship and form the architecture team

- Create an *architecture vision* showing how the architecture contributes to long-term business success to help align the architecture team and gain management sponsorship.
- Develop a *communication plan*. Identify the architecture stakeholders and their communication needs. Plan the activities that will produce the information, and the formats and timeframes in which to communicate the information.

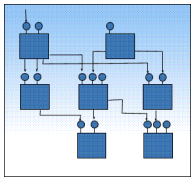
Requirements



Establish and document the architectural requirements

- Establish which *business objectives* apply to the system to ensure that the architecture is aligned with the business agenda.
- Understand the system *context*. Determine the system boundary--what is in scope and what is out of scope. Understand the key organizational, business, competitive, and technical drivers affecting the architecture.
- State the system *value proposition*, establishing how the system will fit the users' agenda and top-level, high-priority goals.
- Document functional requirements by translating user goals into a set of *use cases*.
- Document the *system qualities* or non-functional requirements (e.g., performance and security) by associating them with use cases or creating "what-if" scenarios.

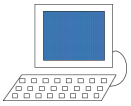
Structuring



Define the architecture

- Based on studying other architectures, and past experience, formulate the *architectural style, concepts, mechanisms* and *principles* that will guide the architecture team during the next steps of structuring.
- Decompose the system into *components.* Identify the *responsibilities* of each component and *interconnections* between components.
- Model the dynamic behavior of the system, using *UML collaboration diagrams* to think through and refine the responsibilities and interfaces of the components.
- Create *component specifications*. Each should include a summary description of services the component provides, the component owner's name, IID and version names, message signatures (IDL), a description of the operations, constraints or pre-post conditions for each operation (these may be represented in a state diagram), the concurrency model, constraints on component composition, a lifecycle model, how the component is instantiated, how it is named, a typical use scenario, a programming example, exceptions, and a test or performance suite.
- Map the components onto the processes of the physical system. Evaluate alternative solutions against requirements such as performance and scaling.

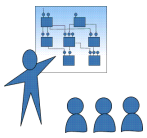
Validation



Validate that the architecture meets the requirements

- Conduct *architecture assessments.* These involve modeling and walking-through scenarios that exemplify requirements, as well as assessment by experts who look for gaps and weaknesses in the architecture based on their experience.
- Develop *prototypes* or proofs-of-concept. Take a skeletal version of the architecture all the way through to implementation to prove out critical aspects of the architecture.

Deployment



Deploy the architecture to the developer community

- Help the developer community understand the architecture and its rationale through *consulting, tutorials* and demos.
- Actively watch for and respond to the need for changes to the architecture. Stay engaged!
- Ensure that the designs and implementations adhere to the architecture by being involved in *design reviews*.

Architects

Architects create and maintain software architectures. Technical responsibilities include: developing an architectural vision; experimenting with alternative architectural approaches; creating models and component specification documents; and validating the architecture against requirements and assumptions. Non-technical activities include envisioning the right architectural approach to the customers problem set given the business objectives of the architect's organization; actively selling the architecture to its various stakeholders; and consulting to and training the developer organization in the use of the architecture.

Training

Bredemeyer Consulting specializes in architecture competency development. We typically work with architecture teams, providing training and mentoring to accelerate the creation or migration of an architecture. We offer a limited number of **Software Architecture** Workshops for open enrollment. The next open enrollment workshops are in:

- San Diego, CA: March 11-14, 2002
- London, UK: April 8-11, 2002

We are also running a 1-day version of our **Role of the Architect** Workshop in San Diego and London. For further information, please call us at (812) 335-1653 or visit **http://www.bredemeyer.com/training.htm**.

BREDEMEYER CONSULTING

TEL: (812) 335-1653 FAX: (812) 335-1652 WEB: http://www.bredemeyer.com