Tutor ~Book

Preview

Introduction This Tutor KnowBook is one of four interrelated ~Books of information that support knowledge workers as they communicate reliably and effectively.

Role	Description
Adviser	Recommends options and guidelines to apply in specific situations
Helper	Helps worker apply KnowSys functions and operations as needed
Librarian	Looks up sources and references needed for work
Tutor	Teaches worker both acting and knowing input for work

Finder 1 You can find support information in the following ~Books as shown below. Click on "Go to Finder 1" near page bottom to return here.

Support	Page	Information
Tutor	3	KnowSys Actions
Tutor	23	KnowSys Job Aids
Adviser	KnowSys seminar guide - Adviser	
Helper	KnowSys seminar guide - Helper	
Librarian	KnowSys seminar guide - Librarian	

Go to Finder 1

Notes

This page is intentionally left blank.

KnowSys Actions

Preview

Introduction

This preview introduces KnowSys actions.

These actions support the Tutor support function for KnowSys — the minimum set of knowledge a worker needs to communicate intelligibly in today's information media, whether paper-based or electronic.

Purpose

KnowSys actions direct workers to build basic units of the KnowSys model, from smallest to largest amount of information —

- KnowBit,
- KnowByte,
- KnowBoot,
- KnowBatch,
- KnowBook, *and*
- KnowBase.

Finder 2

You can find the following information on the pages shown.

Go to Finder 2 Go to Finder 1

Notes

This page is intentionally left blank.

How To Build a KnowBit

Introduction

This ~Byte of information directs the worker to build a KnowBit.

Definition

A *KnowBit* is a visual information field. It has these critical features:

- a tag stating one basic function or source of the ~Bit content,
- logical and visual separation from other ~Bits, *and*
- a body containing information from a single source and with a single function for a worker.

Alias

The alias for a KnowBit is "~Bit". The alias —

- simplifies the proper name by omitting the prefix, "Know-".
- includes a tilde symbol and upper-case "B" to distinguish this special meaning from other meanings of "bit."

Description

A KnowBit looks like a simple, unconventional paragraph.

It shapes less information than any other visual information field.

A ~Bit displays —

- one ~Bit tag in bold style, outdented left from the ~Bit body.
- borders above and below ~Bit body only, separating it from other ~Bits.
- one ~Bit body with —
  - plain style text and font size equal to the ~Bit tag,
  - a stem phrase or sentence.

Note: The stem introduces —

- one to seven sentences of plain style text only, *or*
- a list, a table, *or* graphics.

Purpose

A ~Bit replaces a conventional paragraph with a precise and reusable unit.

More…

More… How To Build a KnowBit

Rules

Three rules govern building a ~Bit.

- Any ~Bit must be built into a ~Byte before publishing it to workers.
- A ~Bit must contain a single unit of information —
    - from a single source, *and*
    - with a single function for workers.
- A ~Bit must contain a stem phrase or stem sentence introducing —
    - one to seven sentences of text, *or*
    - one list, table, or picture.

Consequences

Applying the rules for building a ~Bit has these consequences.

- Reusing such a ~Bit is free from ties to irrelevant information.
- Revising such a ~Bit is less complicated.
- Grasping the meaning of such a ~Bit is faster.
- Building a precise and accurate tag for such a ~Bit is easier.

Guidelines

Whenever possible, express ~Bit information in —

- the present tense.
- the active voice.
- conversational style.
- informal tone.
- a visual display area small enough for worker to view it all at once.

Exceptions

You may use passive voice in the following situations. You —

- do not know who is responsible for the action.
- cannot reasonably list all the parties responsible for the action.
- cannot reveal the responsible party or parties with impunity.
- think that the action is more important than who does it.

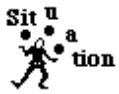
More…

More…How To Build a KnowBit

Applying the rules and guidelines, follow these directions to build a KnowBit.

Note: The order of steps in this action applies to writing done by hand or by word processor.

Step	Action
1	Answer the question, "Who will work with this ~Bit ?"
2	Answer the question, "What will the worker do with this ~Bit ?"
3	Answer the question, "What is the source for this ~Bit?"
4	Write the ~Bit tag to satisfy these three questions.
5	Write the ~Bit body to answer the question, "What is the least content the worker needs?" • *If possible, then* write the ~Bit content in present tense, active voice. • *Otherwise*, write in active voice, past or future tense.
6	Border the ~Bit body above and below with double spacing or lines.
7	Save the ~Bit for later reuse.



Build a ~Bit tag by applying these guidelines with good judgment to the answers for the questions you asked in the earlier action.

Apply these guidelines when building a ~Bit tag.

- State a single function or source for the ~Bit contents.
- Guide the worker to needed information accurately and quickly.
- State specific content about the ~Bit briefly and clearly.

Guideline about brevity

Include just enough words in a ~Bit tag to —

- identify the function or source of the ~Bit contents.
- distinguish this ~Bit from others of similar contents.

Go to Finder 2 Go to Finder 1

How To Build a KnowByte

Introduction

This ~Byte of information directs the worker to build a KnowByte.
It is a self-referring example.

Definition

A KnowByte is a visual information field with three features:

- one ~Bit related to a basic function for a worker,
- one to seven support ~Bits, including an introductory ~Bit, *and*
- a tag stating a basic function or source of the ~Byte content.

Alias

The alias for a KnowByte is "~Byte". The alias —

- simplifies the proper name by omitting the prefix, "Know-".
- is similar to "byte" in programming code in that it may contain eight "bits."
- includes a tilde symbol and upper-case "B" to distinguish this special meaning from the meaning of "byte" as computer programmers know it.

Analogy

As a word adds special meaning to a combination of letters, so also
a KnowByte adds special meaning to a combination of KnowBits.

Description

A KnowByte looks like a simple, but unconventional page of information.
It contains the least amount of *publishable* information of any visual information field. A KnowByte displays —

- a tag at top left in bold style text, two font sizes larger than ~Bit font size.
- two to eight ~Bit tags in bold style, out dented left from ~Bit bodies.
- two to eight ~Bit bodies in plain style text and font size equal to ~Bit tags.
- clear and wide margins at top and left for accessible scanning of all tags.
- borders that separate ~Bits from each other and from the ~Byte tag, without interfering with accessible scanning of marginal tags.

Purpose

A ~Byte replaces a conventional page with a flexible, precise and reusable combination of information ~Bits. Use a ~Byte to publish one or more related ~Bits for quick and accurate access by workers.

More…

More… How To Build a KnowByte

Rules

These rules govern building a ~Byte for display on paper or screen.

- A ~Byte must focus on one basic content, function, or source, expressed in one basic ~Bit.
- A ~Byte may combine one to seven support ~Bits with its basic ~Bit.
- An introductory ~Bit must orient workers to the ~Bits making up a ~Byte.
- Workers must recognize a ~Byte split onto two visual display areas.
- A ~Byte must have a tag and related ~Bit tags that various workers can read accurately and quickly.

Guidelines

These guidelines apply to the rules for building a ~Byte.

- Use a minimum font size of 10 or larger for text of ~Bits within a ~Byte.
- Include a "More..." tag on both borders of a ~Byte split onto two visual display areas.

Consequences

The consequences for applying the rules for building a ~Byte are these:

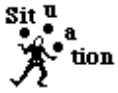
- Reusing a ~Byte is free from ties to irrelevant information.
- Revising a ~Byte is less complicated.
- Scanning a ~Byte is more precise and accurate.

More…

More…How To Build a KnowByte

Applying the rules and guidelines, follow these directions to build a KnowByte. Use the blank KnowByte field as a job aid.

Step	Action
1	Answer the question, "Who will work with this ~Byte?"
2	Answer the question, "What will the worker do with this ~Byte?"
3	Answer the question, "What is the source for this ~Byte?"
4	Build the basic ~Bit that satisfies the first three questions.
5	Answer the question, "What is the least support information the worker needs beyond the basic ~Bit ?"
6	Build the minimal number of support ~Bits.
7	Build an introductory ~Bit.
8	Build a ~Byte tag that satisfies the first three questions.



Build a ~Byte tag by applying these guidelines with good judgment to the answers for the questions you asked in the earlier action.

Apply these guidelines when building a ~Byte tag —

- State the ~Byte's function or source to agree with the basic ~Bit.
- State specific content of the ~Byte briefly and clearly.
- Use a bold-style font, two sizes larger than text of ~Bits.
- Position tag consistently at top left of display area.
- Include just enough words in a ~Byte tag to —
    - identify the function or source of the ~Byte contents.
    - distinguish this ~Byte from others of similar contents.

Go to Finder 2 Go to Finder 1

How To Build a KnowBoot

Introduction

This ~Byte of information directs the worker to build a KnowBoot. The worker will need this information when following the directions for building a KnowBatch.

Definition

A *KnowBoot* is a ~Byte of information that has two metacognitive features: it —

- introduces a worker to one ~Batch, one ~Book, or one ~Base, *and*
- orients a ~Batch, ~Book, or ~Base within a whole system.

Description

A KnowBoot is a cover screen or cover sheet that —

- precedes a ~Batch, ~Book, or ~Base.
- displays a "Preview" ~Byte tag at top left.
- displays a ~Batch, ~Book, or ~Base tag, centered above the ~Byte tag.
- may include one or more ~Bits for systemic orientation.
- concludes with a "Contents" or "Finder" ~Bit, listing —
  - the tags of subsequent ~Bytes, ~Batches, or ~Books, *and*
  - page numbers or location links to find the contents.

Alias

The alias for a KnowBoot is "~Boot". The alias —

- simplifies the proper name by omitting the prefix, "Know-".
- is similar to "boot" in computer operation, in that it initiates the execution of a program unit by processing a "batch" program.
- includes a tilde symbol and upper-case "B" to distinguish this special meaning from the meaning of "boot" as computer operators know it.

More…

More… How To Build a KnowBoot

Purpose

Use a KnowBoot as a cover screen or cover sheet to —

- introduce a worker to a ~Batch, ~Book, or ~Base,
- connect a ~Batch, ~Book, or ~Base to other related units, *and*
- allow a worker to preview its ~Bytes quickly and simply, without details.

Rule

A KnowBoot must appear first in any series of two or more —

- ~Bytes,
- ~Batches,
- ~Books, *or*
- ~Bases.

Consequences

The consequences for applying the rule for building a ~Boot are these:

Workers will —

- fit a ~Batch, ~Book, or ~Base into a recognizable system construct.
- have realistic expectations for a series of information units.
- affirm the relationship of this ~Batch, ~Book, or ~Base to other units.

More…

More… How To Build a KnowBoot

Applying the rule, follow these directions to build a KnowBoot.
Use the blank KnowBoot field as a job aid.

Note: The order of steps in this action applies to writing with a word processing application.

Step	Action
1	Answer the question, "Who will work with this ~Batch, ~Book, or ~Base?"
2	Answer the question, "What will the worker do with this ~Batch, ~Book, or ~Base?"
3	Build a Purpose ~Bit to satisfy these two questions.
4	Answer the question, "What is the source for this ~Batch, ~Book, or ~Base?"
5	Build a Source ~Bit to satisfy this question.
6	Do you have the tags for the table of contents? • *If yes*, then build the basic ~Bit telling where to find the contents. • *If no*, leave table of contents empty until you build the tags.
7	Use "Preview" as the tag for this and every ~Boot.

Go to Finder 2 Go to Finder 1

How To Build a KnowBatch

Introduction

This ~Byte of information directs the worker to build a KnowBatch.

Definition

A *KnowBatch* is a visual information field larger than a ~Byte, with three features:

- a ~Boot that tags and previews the KnowBatch body of information,
- a body combining two to seven ~Bytes from various sources, *and*
- one systemic function for a worker.

Note: Systems thinking is especially important at this level of the KnowSys performance support system.

Analogy

As a word integrates a combination of letters, and a sentence integrates a combination of words, so also a KnowBatch integrates a combination of ~Bytes.

Description

A KnowBatch looks like a simple chapter from an unconventional book. It contains a larger amount of publishable information than a ~Byte.

A KnowBatch displays —

- a KnowBatch body combining two to eight ~Bytes.
- a ~Boot preceding the KnowBatch body of information.

Alias

The alias for a KnowBatch is "~Batch". The alias —

- simplifies the proper name by omitting the prefix, "Know-".
- is similar to "batch" in programming in that a batch of data is processed together.
- includes a tilde symbol and upper-case "B" to distinguish this special meaning from the meaning of "batch" as computer programmers know it.

More…

More… How To Build a KnowBatch

Purpose

Use a ~Batch to publish two-to-seven related ~Bytes for quick and accurate access by workers. A ~Batch replaces a chapter of a conventional book with a combination of ~Bytes which is —

- flexible,
- precise, *and*
- reusable.

Rules

Two rules govern the task of building a ~Batch publishable on paper or screens of a digital device: a —

- ~Boot must always precede the body of a ~Batch.
- ~Batch may have more than one author for its combined ~Bytes.
- ~Batch must express a special function of its combined information at a *systemic* level, rather than at any isolated, singular level.

Consequences

The consequences of applying the rules for building a ~Batch are these:

- Opening a ~Batch with a ~Boot tells workers what is the special function of the combination of ~Bytes in the ~Batch body.

- Positioning a ~Boot first in the order of ~Bytes in a ~Batch allows the worker to preview a ~Batch, and form relevant expectations about its value.

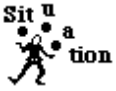
More…

More… How To Build a KnowBatch

Applying the rules, follow these directions to build a KnowBatch.
Use the job aid to help seriate the ~Bytes in the ~Batch in Step 6.

Note: The order of steps in this action assumes that the worker is using a word-processing application.

Step	Action
1	Answer the question, "Who will work with this ~Batch ?"
2	Answer the question, "What will the workers do with this ~Batch?"
3	Answer the question, "What are the sources for this ~Batch?"
4	Build a tentative ~Boot to satisfy these three questions.
5	Build the fewest number of ~Bytes that will satisfy the workers' need for the basic systemic function of this ~Batch.
6	Seriate the ~Bytes to suit the basic systemic function of this ~Batch.
7	Cut and paste ~Byte tags into the ~Boot table of contents.
8	Does the tentative ~Boot still satisfy the first three questions? • *If yes, then* save the ~Batch for publishing to workers. • *If no,* revise the ~Boot before saving ~Batch.



Use your judgment to seriate the combined ~Bytes in a ~Batch that best meets workers' needs.

Note: Some workers will prefer a series of ~Bytes in a ~Batch different from other workers. Validly and reliably tagging all ~Bits and ~Bytes allows workers random access to any information that interests them. The serial order of ~Bytes is then less critical.

Go to Finder 2 Go to Finder 1

How To Build a KnowBook

Introduction

This ~Byte of information directs the worker to build a KnowBook.

Definition

A *KnowBook* is a visual information field which combines two to seven ~Batches with different functions and sources. A KnowBook has these three features:

- a ~Boot that tags and previews the KnowBook body of information,
- a body combining two to seven ~Batches from various sources, *and*
- one basic systemic function for a worker.

Note: Systems thinking is especially important at this level of the KnowSys performance support system.

Description

A KnowBook looks like a simple, unconventional book. It contains a larger amount of publishable information than a ~Batch.

A KnowBook displays —

- a KnowBook body combining two to seven ~Batches.
- a ~Boot preceding the KnowBook body of information.

Alias

The alias for a KnowBook is "~Book". The alias —

- simplifies the proper name by omitting the prefix, "Know-".
- is similar to "book" in conventional publishing, in that it combines ~Batches to replace conventional "chapters" about a subject of interest.
- includes a tilde symbol and upper-case "B" to distinguish this special meaning from the meaning of "book" as editors and publishers know it.

More…

More… How To Build a KnowBook

Purpose

Use a ~Book to publish two to seven related ~Batches for quick and accurate access by workers. A ~Book replaces a conventional book with a combination of ~Batches which is —

- flexible,
- precise, *and*
- reusable.

Rules

Two rules govern building a ~Book publishable on paper or personal computer screens: a —

- ~Boot must always precede the body of a ~Book.
- ~Book must express a special function of its combined information at a *systemic* level, rather than at any isolated, singular level.

Consequences

The consequences for applying the rules for building a ~Book are these:

- Opening a ~Book with a ~Boot tells workers what is the special function of the combination of ~Batches in the ~Book body.
- Positioning a ~Boot first in the order of ~Batches in a ~Book allows the worker to preview a ~Book, and form relevant expectations about its value.

More…

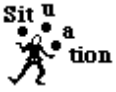
More… How To Build a KnowBook

Applying the rules, follow these directions to build a ~Book.

Use the job aid to help seriate the ~Batches in the ~Book in Step 6.

Note: The order of steps in this action assumes that workers are using word processors.

Step	Action
1	Answer the question, "Who will work with this ~Book ?"
2	Answer the question, "What will the workers do with this ~Book?"
3	Answer the question, "What are the sources for this ~Book?"
4	Build a tentative ~Boot to satisfy these three questions.
5	Build the fewest number of ~Batches that will satisfy the workers' need for the basic systemic function of this ~Book.
6	Seriate the ~Batches to suit the basic systemic function of this ~Book.
7	Cut and paste ~Batch tags into the ~Boot table of contents.
8	Does the tentative ~Boot still satisfy the first three questions? • *If yes, then* save the ~Book for publishing to workers. • *If no,* revise the ~Boot before saving ~Book.



Use your judgment to seriate the combined ~Batches in a ~Book that best meets workers' needs.

Note: Some workers will prefer a series of ~Batches in a ~Book different from other workers.
Validly and reliably tagging all ~Bits, ~Bytes, and ~Batches allows workers random access to any information that interests them.
The serial order of ~Bytes or ~Batches is then less critical.

Go to Finder 2 Go to Finder 1

How To Build a KnowBase

Introduction

This ~Byte of information directs the worker to build a KnowBase.

Definition

A *KnowBase* is a visual information field which combines two to seven ~Books related to a basic function. A KnowBase has these three features:

- a ~Boot that tags and previews the KnowBase body of information,
- a body combining two to seven ~Books from various sources, and
- one basic systemic function for a worker.

Note: Systems thinking is especially important at this level of the KnowSys performance support system.

Description

A KnowBase looks like an unconventional encyclopedia of knowledge about a limited domain. It contains a larger amount of publishable information than a ~Book.

A KnowBase displays —

- a ~Boot preceding the KnowBase body of information.
- a KnowBase body combining two to seven ~Books.

Alias

The alias for a KnowBase is "~Base". The alias —

- simplifies the proper name by omitting the prefix, "Know-".
- is similar to "data base" in computer science, in that the knowledge base combines all units of information required to support a knowledge system.
- includes a tilde symbol and upper-case "B" to distinguish this special meaning from the meaning of "base" as computer scientists know it.

More…

More… How To Build a KnowBase

Purpose

Use a ~Base to publish two to seven related ~Books for quick and accurate access by workers. A ~Base replaces a conventional encyclopedia of books with a combination of ~Books which is —

- flexible,
- precise, *and*
- reusable.

Rules

Two rules govern building a ~Base publishable on paper or personal computer screens.

- A ~Boot must always precede the body of a ~Base.
- A ~Base must express a special function of its combined information at a *systemic* level, rather than at any isolated, singular level.

Consequences

Applying the rules for building a ~Base has two consequences:

- Opening a ~Base with a ~Boot tells workers what special function derives from the combination of ~Books in the ~Base body.
- Positioning a ~Boot first in the order of ~Books in a ~Base allows the worker to preview a ~Base, and form relevant expectations about its value.

More…

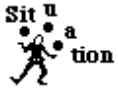
More… How To Build a KnowBase

Applying the rules, follow these directions to build a ~Base.

Use the job aid to help seriate the ~Books in the ~Base in Step 6.

Note: The order of steps in this action assumes that workers are using word processors.

Step	Action
1	Answer the question, "Who will work with this ~Base ?"
2	Answer the question, "What will the workers do with this ~Base?"
3	Answer the question, "What are the sources for this ~Base?"
4	Build a tentative ~Boot to satisfy these three questions.
5	Build the fewest number of ~Books that will satisfy the workers' need for the basic systemic function of this ~Base.
6	Seriate the ~Books to suit the basic systemic function of this ~Base.
7	Cut and paste ~Book tags into the ~Boot table of contents.
8	Does the tentative ~Boot still satisfy the first three questions? • *If yes, then* save the ~Base for publishing to workers. • *If no, then* revise the ~Boot before saving ~Base.



Use your judgment to seriate the combined ~Books in a ~Base that best meets workers' needs.

Note: Some workers will prefer a series of ~Books in a ~Base different from other workers.
Validly and reliably tagging all ~Bits, ~Bytes, ~Batches, and ~Books allows workers random access to any information that interests them. The serial order of ~Books is then less critical.

Go to Finder 2 Go to Finder 1

KnowSys Job Aids

Preview

Introduction

This preview introduces KnowSys job aids.

These job aids support the Tutor support function for KnowSys — the minimum set of knowledge a worker needs to communicate intelligibly in today's information media, whether paper-based or electronic.

Purpose

KnowSys job aids give workers guidelines and blank fields for —

- judging a situation.
- building basic units of the KnowSys model, from smallest to largest amount of information –
    - KnowBit,
    - KnowByte,
    - KnowBoot,
    - KnowBatch,
    - KnowBook, *and*
    - KnowBase.

Finder 3

You can find the following information on the pages shown.

Go to Finder 3 Go to Finder 2 Go to Finder 1

Job Aid—Judging a Situation

Introduction

Judging a situation means judging whether one or more options best satisfies the standards which measure your quality values.

Principles

These principles govern your act of judgment.

You should —

- match options directly to guidelines for related values.
- use practical guidelines to apply standards indirectly.
- distill guidelines from many experiences of workers in similar situations.

Situation

List your options opposite one or more of the standards in this table. Judge which option best satisfies your values.

Option	Value	Guideline
[A] [B] [C]	Validity	• Listen actively to voices of workers. • Analyze worker needs for documents. • Design a documentation system that meets needs of workers. • Create structured displays of hypermedia.
[A] [B] [C]	Reliability	• Adopt clear standards for display of information fields. • Educate workers to use standards. • Reward workers to use standards.
[A] [B] [C]	Timeliness	• Prioritize tasks systemically. • State systemic consequences of tasks. • Reward workers for timely tasks.
[A] [B] [C]	Robustness	• Analyze interactions of system factors. • Minimize variability between controllable and uncontrollable factors.
[A] [B] [C]	Maintain-ability	• Store structured documents on-line. • Display structured documents in hypermedia.
[A] [B] [C]	Traceability	• Listen actively to voices of workers. • Reinforce worker needs at each phase. • Monitor phase results against cost-effective standards. • Document version data in footers.
[A] [B] [C]	Durability	• Listen actively to voices of workers. • Apply Pareto analysis to select parts that need improving most.
[A] [B] [C]	Aesthetics	• Listen actively to voices of workers. • Scan horizon for better designs.

Go to Finder 3

Job Aid — How To Build a KnowBit

Introduction

This job aid directs and guides you to build valid and reliable ~Bits.
Scan the ~Bit tags in left margin, then decide which ~Bit body to build.

[If introducing a ~Byte or ~Boot, then —

- state where it fits into a system construct,
- describe realistic expectations for the series of information units, *or*
- state how this ~Byte, ~Batch, ~Book, or ~Base relates to other units.]

Picture

[If working with a construct, then put graphics, diagram, picture, or photo in ~Bit body, after a stem phrase or sentence.]

Process

[If working with a system, then put phases in ~Bit body after a stem phrase or sentence. Phases may be linear, branching, looping, or a combination.]

Purpose

[If worker needs incentive to do a task, then use text in ~Bit body. Tell the worker why doing a task adds value.]

Definition

[If working with a concept, then put definition in ~Bit body, after a stem phrase or sentence which emphasizes the concept tag.]

Example

[If workers must recognize an instance of a concept with accuracy, then put one or two examples in ~Bit body, after a stem phrase or sentence.]

Exception

[If a concept may confuse workers, then put one or two exceptions in ~Bit body, after a stem phrase or sentence.]

Description

[If the worker needs a "word picture" of a static or dynamic construct, then describe in ~Bit body what the worker experiences through the senses.]

Policy

[If working with a policy, then put a statement, example, or exception in ~Bit body, after a stem phrase or sentence.]

More…

More…*Job Aid*—How To Build a KnowBit

Rules

Two rules govern building the body of a "Rules" ~Bit.

- Always introduce ~Bit body with a stem phrase or sentence.
- Use the least information required to satisfy needs of workers.

[If working with rules, then put positive statement, example, or exception in ~Bit body, after a stem phrase or sentence.]

Guidelines

[If workers need guidelines to support a rule or policy, then list in ~Bit body what an expert, coach, or mentor would advise the worker to do to comply.]

Exception

[If working with allowable exceptions to a policy or rule, then put an exception in ~Bit body, after a stem phrase or sentence.]

Consequences

[If working with consequences for policy or rule, then put positive statement, example, or exception in ~Bit body, after a stem phrase or sentence.]

Analogy

[If working with a metaconcept, then put comparison of known relationships to comparison of unknown relationships in ~Bit body, after a stem phrase or sentence.]

Metaphor

[If working with a metaconcept, then put an imaginative comparison between members of different classes in ~Bit body, after a stem phrase or sentence — **don't** use the words "as" or "like."]

Simile

[If working with a metaconcept at a **surface** level, then put an imaginative comparison between members of different classes in ~Bit body, after a stem phrase or sentence — use the words "as" or "like."]

Action

[If the worker needs directions to do a task, then state the goal of the task in the stem phrase or sentence.
Combine in the ~Bit body the linear, branching, or looping directions.]

Situation

[If the worker needs to apply guidelines to judge the best handling of a situation, then list the guidelines in the ~Bit body. State in the stem phrase or sentence the directions for judging the situation against related values.]

Go to Finder 3 Go to Finder 2 Go to Finder 1

Job Aid—How To Build a KnowByte [~Byte tag]

Introduction

This blank ~Byte serves as a job aid when considering which support ~Bits to combine with the basic ~Bit.

[Build the basic ~Bit first before the support ~Bits,

including an introductory ~Bit like this one.

- State where this ~Byte fits into a system construct,
- Describe realistic expectations for the series of information units, *or*
- State how this ~Byte relates to other units.]

Rules

[Position earlier than any ~Bit that needs support.]

These rules govern building a ~Byte for display on paper or screen.

- A ~Byte must focus on one basic content, function or source, expressed in one basic ~Bit.
- A ~Byte may combine one to seven support ~Bits with its basic ~Bit.
- An introductory ~Bit must orient workers to the ~Bits making up a ~Byte.
- Workers must recognize a ~Byte split onto two visual display areas.
- A ~Byte must have a tag and related ~Bit tags that various workers can read accurately and quickly.

Guidelines

[Position later than a ~Bit that contains a rule or policy needing support.]

- Use the fewest support ~Bits required to satisfy needs of workers.
- Change the order of ~Bits as desired.
- Build the ~Byte tag last.

Picture

[Position early in ~Bit order when working with a construct.]

Process

[Position early in ~Bit order when working with a system.]

Purpose

[Position early in ~Bit order when the worker needs incentive to do a task.]

More…

More… *Job Aid*—How To Build a KnowByte

Definition	[Position early in ~Bit order when working with a concept.]
Example	[Position later than a definition ~Bit when working with a concept. If supporting a principle ~Bit, position example(s) later.]
Exception - concept	[Position later than an example ~Bit when working with a concept. If supporting a principle, position exception ~Bit later than example(s).]
Description	[Position early in ~Bit order when the worker needs a "word picture."]
Policy	[Position early in ~Bit order when the worker is working with a principle.]
Exception - rule	[Position later than a ~Bit that contains a rule or policy needing support.]
Consequences	[Position later than a ~Bit that contains a rule or policy needing support.]
Analogy	[Position later than a ~Bit that contains a concept.]
Metaphor	[Position later than a ~Bit that contains a concept.]
Simile	[Position later than a ~Bit that contains a concept.]
Action	[Position late in ~Bit order when the worker needs knowledge to do a task.]
Situation	[Position late in ~Bit order when the worker needs guidelines to apply, using judgment.]

Go to Finder 3 Go to Finder 2 Go to Finder 1

Job Aid—How To Build a KnowBoot

Preview *[This tag always identifies a KnowBoot.]*

Introduction

This blank ~Byte serves as a job aid when building a ~Boot to introduce a ~Batch, ~Book, or ~Base.

[Relate this ~Batch, ~Book, or ~Base to other associated units of information.]

Purpose

[State in this support ~Bit why the worker should use this ~Batch, ~Book, or ~Base.]

Source

[State in this support ~Bit the basic source of this ~Batch, ~Book, or ~Base.]

Finder

[Copy into this table or parallel list the tags of the two-to-eight largest visual information fields which follow this KnowBoot, then insert their page numbers or location links.]

You can find the following information on the pages shown.

Page	Information
2	[tag]
3	[tag]
4	[tag]
5	[Etc.]

Go to Finder 3 Go to Finder 2 Go to Finder 1

Job Aid—How To Build a KnowBatch

Introduction

This job aid supports the task of building a KnowBatch in the series that best suits workers. It depends on —

- managing the six basic resources needed by an effective process, *and*
- seriating ~Bytes in relation to a process's basic resource.

Directions

Apply the rules and follow the steps in the Action ~Bit of the Actions file, How To Build a ~Batch.

At Step 6, seriate the ~Bytes within the ~Batch body by —

- focusing on one of six resources in the fishbone picture as basic.
- matching that basic resource to the series recommended by the Resource Series Table.

Note: Validly and reliably tagging of all ~Bits and ~Bytes allows workers random access to any information that interests them most. Therefore, no totally wrong serial order of ~Bytes is possible — only a less efficient one.

Definition: seriating

Seriating is a conceiving skill that orders subjects, objects, or events according to some consistent attribute.

Seriating is one of three thinking skills whose purpose is knowledge acquisition: observing, perceiving, and conceiving.

Table of seriating skills

This table describes the tags that workers use to discriminate among four common methods of seriating.

When seriating by…	then worker tags the skill as… .
units of time	sequencing
repeated construct	patterning
personal values	prioritizing
any other attribute	ordering

More…

More… Job Aid—How To Build a KnowBatch

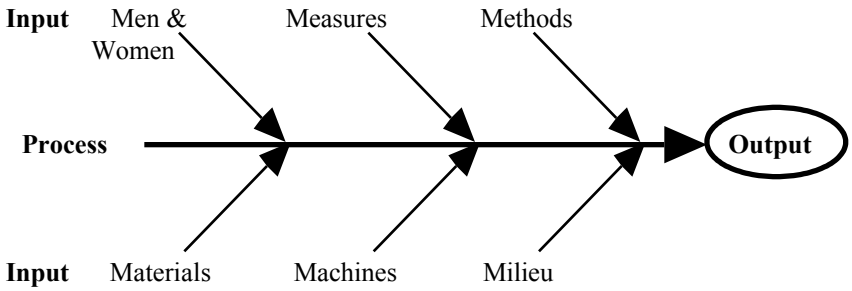
Purpose

Managers and workers often use a fishbone construct to allocate the basic resources that a process needs to "output" a valuable product or service to a paying customer.

Note: You may also use a fishbone construct in reverse order to isolate the causes for a defective product or service.

Picture

This picture displays a "fishbone" construct for managing the basic resources within a quality system. Each "rib" of the process "spine" represents one basic resource category. The "fishhead" is the output of the process.



More…

More…Job Aid—How To Build a KnowBatch

Resource series table

This table displays a recommended series for each resource shown on the fishbone construct, and gives a matching example.

Resource	Recommended Series	Example
men & women	1. system constructs 2. system processes 3. system principles 4. system concepts 5. system facts	Seriated ~Bytes within a personnel system — trees of organizational positions are the focus.
measures	1. system processes 2. system principles 3. system concepts 4. system facts	Seriated ~Bytes within an evaluation system —principles, formulas, and equations are the focus.
methods	1. system processes 2. system constructs 3. system actions & situations 4. system principles 5. system concepts 6. system facts	Seriated ~Bytes within a support service system — individual actions and situations are the focus of the system processes and constructs
materials	1. system processes 2. system constructs 3. system principles 4. system facts	Seriated ~Bytes within a manufacturing system in which the sequence of system phases is as important as system constructs
machines	1. system processes 2. system constructs 3. system actions & situations 4. system principles 5. system facts	Seriated ~Bytes within an operating system — focus is on individual actions and situations involving machines and computers
milieu	1. system processes 2. system constructs 3. system principles 4. system concepts 5. system facts	Seriated ~Bytes within an environmental system — processes are the focus.

Go to Finder 3 Go to Finder 2 Go to Finder 1

Job Aid—How To Build a KnowBook

Introduction

This job aid supports the task of assembling the ~Batches of a KnowBook in the series that best suits workers. It depends on

- managing the six basic resources needed by an effective process,
- seriating ~Batches in relation to a process's basic resource.

Directions

Apply the rules and follow the steps in the Action ~Bit of the Actions file, How To Build a ~Book.

At Step 6, seriate the ~Batches within the ~Book body by —

- focusing on one of six resources in the fishbone picture as basic.
- matching that basic resource to the series recommended by the Resource Series Table.

Note: Validly and reliably tagging of all ~Bits, ~Bytes, and ~Batches allows workers random access to any information that interests them most. Therefore, no totally wrong serial order of ~Bytes and ~Batches is possible—only a less efficient series.

Definition: seriating

Seriating is one of three thinking skills whose purpose is knowledge acquisition: observing, perceiving, and conceiving.
Seriating is a conceiving skill that orders subjects, objects, or events according to some consistent attribute.

Table of seriating skills

This table describes the tags that workers use to discriminate among four common methods of seriating.

When seriating by…	then worker tags the skill as… .
units of time	sequencing
repeated construct	patterning
personal values	prioritizing
any other attribute	ordering

More…

More… Job Aid—How To Build a KnowBook

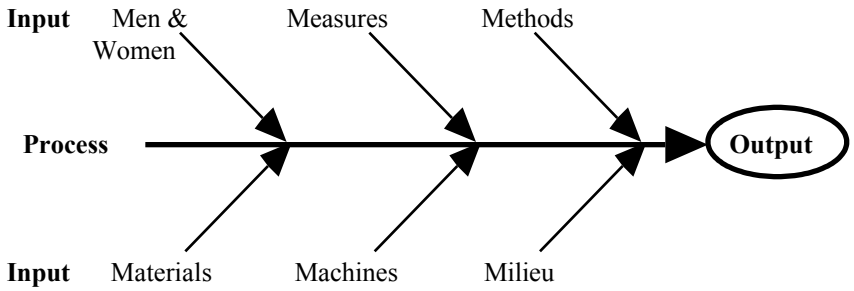
Purpose

Managers and workers often use a fishbone to allocate the basic resources needed for a process which puts out a product or service to a paying customer.

Note: You may also use it in reverse order to isolate the causes for a defective product or service.

Picture

This picture displays a "fishbone" construct for managing the basic resources within a quality system. Each "rib" of the process "spine" represents one basic resource category. The "fishhead" is the output of the process.



More…

More… Job Aid—How To Build a KnowBook

Resource series table

This table displays a recommended series for each resource shown on the fishbone construct, and gives a matching example.

Resource	Recommended Series	Example
men & women	1. system constructs 2. system processes 3. system principles 4. system concepts 5. system facts	Seriated ~Batches within a personnel system — trees of organizational positions are the focus.
measures	1. system processes 2. system principles 3. system concepts 4. system facts	Seriated ~Batches within an evaluation system —principles, formulas, and equations are the focus.
methods	1. system processes 2. system constructs 3. system actions & situations 4. system principles 5. system concepts 6. system facts	Seriated ~Batches within a support service system — individual actions and situations are the focus of the system processes and constructs
materials	1. system processes 2. system constructs 3. system principles 4. system facts	Seriated ~Batches within a manufacturing system in which the sequence of system phases is as important as system constructs
machines	1. system processes 2. system constructs 3. system actions & situations 4. system principles 5. system facts	Seriated ~Batches within an operating system — focus is on individual actions and situations involving machines and computers
milieu	1. system processes 2. system constructs 3. system principles 4. system concepts 5. system facts	Seriated ~Batches within an environmental system — processes are the focus.

Go to Finder 3 Go to Finder 2 Go to Finder 1

Job Aid—How To Build a KnowBase

Introduction

This job aid supports the task of assembling the ~Books of a KnowBase in the series that best suits workers. It depends on

- managing the six basic resources needed by an effective process,
- seriating ~Books in relation to a process's basic resource.

Directions

Apply the rules and follow the steps in the Action ~Bit of the Actions file, How To Build a ~Base.

At Step 6, seriate the ~Books within the ~Base body by —

- focusing on one of six resources in the fishbone picture as basic.
- matching that basic resource to the series recommended by the Resource Series Table.

Note: Validly and reliably tagging of all ~Bits, ~Bytes, ~Batches, and ~Books allows workers random access to any information that interests them most. Therefore, no totally wrong serial order of ~Books is possible—only a less efficient series.

Definition: seriating

Seriating is one of three thinking skills whose purpose is knowledge acquisition: observing, perceiving, and conceiving.
Seriating is a conceiving skill that orders subjects, objects, or events according to some consistent attribute.

Table of seriating skills

This table describes the tags that workers use to discriminate among four common methods of seriating.

When seriating by…	then worker tags the skill as… .
units of time	sequencing
repeated construct	patterning
personal values	prioritizing
any other attribute	ordering

More…

More…Job Aid—How To Build a KnowBase

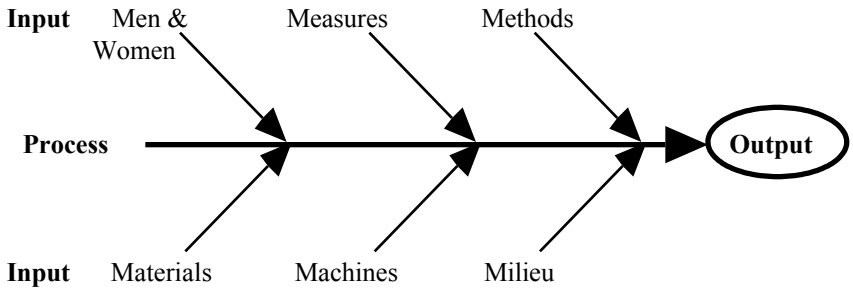
Purpose

Managers and workers often use a fishbone to allocate the basic resources needed for a process which puts out a product or service to a paying customer.

Note: You may also use it in reverse order to isolate the causes for a defective product or service.

Picture

This picture displays a "fishbone" construct for managing the basic resources within a quality system. Each "rib" of the process "spine" represents one basic resource category. The "fishhead" is the output of the process.



More…

More… Job Aid—How To Build a KnowBase

Resource series table

This table displays a recommended series for each resource shown on the fishbone construct, and gives a matching example.

Resource	Recommended Series	Example
men & women	1. system constructs 2. system processes 3. system principles 4. system concepts 5. system facts	Seriated ~Books within a personnel system — trees of organizational positions are the focus.
measures	1. system processes 2. system principles 3. system concepts 4. system facts	Seriated ~Books within an evaluation system —principles, formulas, and equations are the focus.
methods	1. system processes 2. system constructs 3. system actions & situations 4. system principles 5. system concepts 6. system facts	Seriated ~Books within a support service system — individual actions and situations are the focus of the system processes and constructs
materials	1. system processes 2. system constructs 3. system principles 4. system facts	Seriated ~Books within a manufacturing system in which the sequence of system phases is as important as system constructs
machines	1. system processes 2. system constructs 3. system actions & situations 4. system principles 5. system facts	Seriated ~Books within an operating system — focus is on individual actions and situations involving machines and computers
milieu	1. system processes 2. system constructs 3. system principles 4. system concepts 5. system facts	Seriated ~Books within an environmental system — processes are the focus.

Go to Finder 3 Go to Finder 2 Go to Finder 1