Proceedings of the Second International Workshop On Agent-Oriented Software Engineering (AOSE-2001), Montreal, Canada, May 29th 2001.

Determining When to Use an Agent-Oriented Software Engineering Paradigm

Scott A. O'Malley[1] and Scott A. DeLoach[2]

[1]Department of Electrical and Computer Engineering, Air Force Institute of Technology
Wright-Patterson Air Force Base, Ohio 45433-7765
omalleys@stratcom.mil

[2]Department of Computing and Information Sciences, Kansas State University
212 Nichols Hall, Manhattan, KS 66506
sdeloach@cis.ksu.edu

Abstract. With the emergence of agent-oriented software engineering techniques, software engineers have a new way of conceptualizing complex distributed software requirements. To help determine the most appropriate software engineering methodology, a set of defining criteria is required. In this paper, we describe out approach to determining these criteria, as well as a technique to assist software engineers with the selection of a software engineering methodology based on those criteria.

1 Introduction

Software engineers have a number of options when it comes to developing solutions for complex, distributed software requirements. One emerging technique is the development of multiagent systems. There are a number of reasons a software developer may consider a multiagent system. In particular, multiagent systems can provide benefits such as processing speed-up, reduced communication bandwidth, and increased reliability [10]. However, the academic community, as well as industry, is still trying to determine which problems call for a multiagent approach [8, 11].

Once a designer has made the decision to use a multiagent design, a number of methodologies exist for building multiagent systems [2, 3, 4, 7, 14, 15, 19]. The methodologies range from extensions of existing object-oriented methodologies to new *agent-oriented techniques*, which offer a new perspective to developing multiagent systems by increasing the level of abstraction the developer uses to analyze and design the system. As agent-oriented software engineering techniques are becoming more popular, software engineers must select the particular approach that is best suited for the problem they are solving.

Our research at the Air Force Institute of Technology has focused on providing software engineers and managers with a decision-making framework to determine an appropriate methodology when faced with a set of viable software engineering methodology alternatives [12]. This paper focuses on the method we applied for

developing this framework. The primary challenge in developing this framework was selecting a valid set of criteria upon which to base the decision.

The remainder of this section addresses other approaches to determining when an agent-oriented approach is appropriate as well as techniques for classifying software engineering methodologies. Section 2 describes the process we used to define the criteria for the decision-making framework. Section 3 describes a survey that we conducted in November and December 2000 to validate that criteria. Section 4 discusses the results of the survey. Section 5 provides the context in which we applied the criteria to the decision-making framework. Finally, Section 6 presents our conclusions.

1.1 Related Techniques

The strategy taken by Jennings and Wooldridge was to provide "intellectual justification" [8] for the validity of the agent-oriented techniques. Their justification, however, comes from a qualitative analysis of how well the technique addresses the principles that allow software engineering techniques to deal with complex problems proposed by Booch: abstraction, decomposition, and hierarchy [1, 8]. They leave "understanding of the situations in which agent solutions are appropriate" as an outstanding issue [8].

The European Institute for Research and Strategic Studies in Telecommunications (EURESCOM) used a different strategy in 1999 when they began a project to explore the use of agent technologies within the European telecommunications industry. One of the project's three objectives is to "define guidance for the identification of application areas where an agent-based approach is better suited than other approaches" [11]. The consortium produced the following five guidelines to help a developer decide whether an agent-oriented approach is appropriate [11]:

1. An agent-oriented approach is beneficial in situations where complex/diverse types of communication are required.
2. An agent-oriented approach is beneficial when the system must perform well in situations where it is not practical/possible to specify its behavior on a case-by-case basis.
3. An agent-oriented approach is beneficial in situations involving negotiation, co-operation and competition among different entities.
4. An agent-oriented approach is beneficial when the system must act autonomously.
5. An agent-oriented approach is beneficial when it is anticipated that the system will be expanded, modified or when the system purpose is expected to change.

These guidelines are a good beginning in determining whether or not an agent-oriented approach is well suited to a particular problem. However, based on these guidelines alone, there is still no clear answer.

1.2 Software Engineering Methodology Classification

In 1988, the Software Engineering Institute (SEI) presented a set of guidelines for assessing software development methods for real-time systems [17]. The guidelines define a five-step process for evaluating different methodologies. These five steps are:

1. Needs Analysis – Determine the important characteristics of the system to be developed and how individual methods help developers deal with those characteristics.
2. Constraint Identification – Identify the constraints imposed on the permitted solutions and determine how individual methods help developers deal with those constraints.
3. User Requirements – Determine the general usage characteristics of the individual methods.
4. Management Issues – Determine the support provided by the method to those who must manage the development process as well as the costs and benefits of adopting and using the method.
5. Introduction Plan – Develop an understanding of the issues that the method does not address and a plan to augment the method in those areas where it is deficient.

Based on these steps, the consortium developed questions to help analyze prospective methodologies. Some of the questions are meant to be rhetorical, while others require an in-depth knowledge of the methodology and its representations. The purpose of the questions is to make the assessor form an opinion regarding the methodology; however, this process does not over-simplify the problem of selecting a methodology. The questions do provide a framework to present a systematic evaluation process.

We based our assessment process on SEI's existing work in classifying software methods, which includes three major areas of characterization [6]. The SEI process involves determining what a method is, what a method does, and what issues the method addresses. SEI's three areas of characterization are:

- Technical Characteristics
- Management Characteristics
- Usage Characteristics

The *Technical Characteristics* look at classifying the technical characteristics of the software development through the three stages of development (specification, design, and implementation). The characteristics of the software problem dealt with during the specification—or analysis—phase relate to the behavioral and functional views of the problem. These views are carried through to the other stages of the system development. During the design phase, the behavioral and functional views are mapped into the behavioral and functional characteristics of the function. Effective methods allow for smooth transition across these stages and allow the ability to trace functional and behavioral characteristics through all stages of development.

The next set of characteristics, *Management*, is important for considering the support that a method provides to management when evaluating different methods.

The characterization should consider how well the method deals with typical management and project issues such as estimating, planning and reviewing. The characterization should also look at how the method is related to the needs and processes that exist within the organization. Management practices are often a difficult thing to change and identifying potential changes is an important factor in adopting a new methodology [6].

The third set, *Usage Characteristics*, captures and describes the characteristics of the methodology that will affect its use by an organization. These characteristics include the basis for the methodology, the availability of training, and the availability of tool support. This characterization is important in understanding the magnitude of change involved with selection of a methodology.

2 Defining Decision Criteria

The challenge of selecting an appropriate methodology for a software development project is in understanding the differences between the methodologies. The ability to classify these methodologies is crucial to the understanding.

With the characteristics developed in [6] in mind, a set of criteria was developed. For the framework defined in [12], we combined the management and usage characteristics into one category, called *Management Issues*. The technical characteristics of the methodology are captured in the *Program Requirements category*. Each of these categories is discussed in detail below.

2.1 Management Issues

As indicated above, this category is closely related to the management and usage characteristics as defined by [6]. Because of their universal applicability, many of the issues addressed that pertain to this category are taken from the [17], as the management and usage issues for selecting a software development method for real-time systems are practical for any type of system. Below is the initial set of issues selected for this category.

- Cost of Acquiring the Methodology (Meth)
- Cost of Acquiring Support Tools (Tool)
- Availability of Reusable Components (Reuse)
- Effects on Organizational Business Practices (Org)
- Compliance with Standards (Stan)
- Traceability of Changes (Chan)

The first two issues deal with costs involved with selecting the methodology. Specifically, *Cost of Acquiring the Methodology* involves the costs associated with adopting the methodology for use. Factors that influence this issue include the costs incurred by sending personnel to available training, the purchase of reference material, etc. Additionally, *Cost of Acquiring Support Tools* deals with the costs incurred by purchasing tools that support the methodology. The tools include CASE

tools as well as programming development tools. Further, the cost of factors such as additional hardware/software to operate the tools, maintenance costs for the tools, and training, should be included.

Another issue that indirectly deals with cost is *Availability of Reusable Components*. The incorporation of previously developed software into a new system reduces the overall design, implementation, and testing phases for software development. This category is used to measure the methodology's ability to incorporate predefined components into the system.

The final three issues reflect usage issues. First, *Effects on Organizational Business Practices* measures the impact the adoption of a methodology will have on the existing business practices of the organization. The business practice includes ideas such as tracking development progress through milestones, reports, and customer interactions. Next, *Compliance with Standards* is proposed to determine how well an alternative is able to meet standards, whether local to the organization or outside the organization such as national or international. Finally, the last issue in this category, *Traceability of Changes*, measures the methodology's support to trace changes throughout the development lifecycle.

2.2 Project Requirements

The second category of criteria, Project Requirements, is related to the technical characteristics. For this category, the criteria for real-time systems are not directly relevant. In order to derive a set of criteria, we turned to current research and identified a number of technical issues that relate to complex software systems [10, 16]. The issues selected are:

- Legacy System Integration (Leg)
- Distribution (Dis)
- Environment (Env)
- Dynamic System Structure (Struc)
- Interaction (Int)
- Scalability (Scal)
- Agility and Robustness (Agi)

The first three issues in this category relate to constraints of the problem. First, *Legacy System Integration* is a measurement of the methodology's ability to support for the incorporation of previously developed systems, commonly called legacy systems, with the new project requirement. Next, *Distribution* focuses on the ability to support the modeling of distributed aspects of the problem. Distribution can occur in the form of processors, resources, or information. Then, *Environment* measures the methodology's support of developing software systems for environments that have heterogeneous hardware or software.

The next three issues in the category are *Dynamic System Structure, Scalability*, and *Agility and Robustness*. *Dynamic System Structure* represents the methodology's ability to develop software capable of handling the introduction and removal of system components in a manner that is not detrimental to the users of the system is considered in this category. *Scalability*, similar to *Dynamic System Structure*,

measures the methodology's ability to develop software capable of handling the introduction and removal of system-level resources while minimizing the impact on users. Last, *Agility and Robustness* focuses on the methodology's ability to create flexible software systems that will be resilient to dynamic changes in the environment.

The final issue in the Project Requirements category is *Interaction.* This category determines the methodology's ability to handle the interaction between system-level components as well as entities outside the system such as human users and other systems.

3 Survey

After we selected the criteria above based on a number of literature sources [11, 16], the compiled list was presented to software engineering professionals in academia, industry, and government through a survey questionnaire on the Internet for validation [12]. The purpose of the survey was to collect the opinions of software engineering practitioners with regard to the importance of each of the evaluation considerations to the overall decision.

In order to increase survey participation, an announcement was distributed to software engineering professionals through electronic mail lists maintained by the Object Management Group (OMG), University of Maryland Agent Web, and the Software Engineering Research Network at the University of Calgary. In addition to these broadcast mailings, announcements requesting participation were placed on related, moderated newsgroups—comp.ai and comp.software-eng. Finally, requests were sent directly to a number of respected academics, researchers, and industry leaders.

3.1 Survey Analysis

The period for response collection was set at three weeks. Over that period, thirty-three valid responses were collected. The survey began with some basic demographic questions in order to develop a profile of the responders. Of the thirty-three responders, twenty-two people indicated that they were associated with the academic community, three responders were associated with government organizations, and eight were associated with the industrial/commercial sector. As for experience, seventeen indicated 1-5 years of experience in their field. Nine responders categorized themselves as having 5-10 years of experience, and seven responders indicated over 10 years of experience.

The survey also collected the opinions of the responders on the importance of the evaluation consideration that were proposed for the decision as well as their thoughts on the suggested factors, the relative weighting of the management and technical categories, and additional possible factors. As for the criteria proposed, the responders were asked to rate each on a scale of zero to four. Additionally, responders could leave considerations "not rated".

The set of scores each factor received indicates that the responders believed the technical issues are more important that the management issues. Fig. 1 shows the average scores each of the considerations received. Again, the responders felt more emphasis should be placed on technical issues versus management issues.

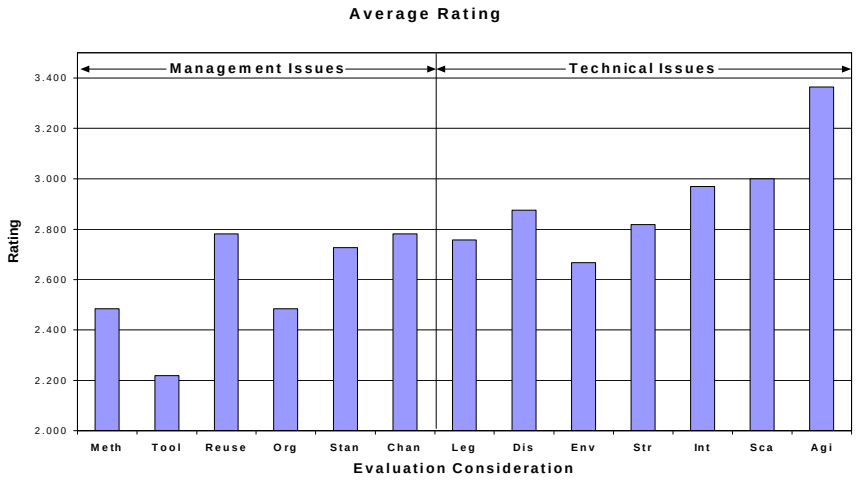

Fig. 1. Average rating for proposed evaluation considerations

The survey asked whether basing the weights for the evaluation considerations relative to only the other considerations in the same category was more appropriate than determining weights relative to all of the considerations. The majority of responses were to determine weights relative to all of the considerations. Most responders did provide an opinion on the total weight each of the major issues. Like the trend seen in Fig. 1, fourteen responders felt that the technical issues should influence the decision more than the management issues. On the other hand, five responders felt that the management issues should weigh more on the decision. Three responders indicated that both sets of issues should have an equal weight. The remaining responders did not specify a particular partitioning. Table 1 shows the data gathered from this particular question.

Finally, the survey posed the question: what important factors are missing? Several alternatives were suggested for the cost category. Responders indicated that other factors would have more significance to the problem such as a cost/benefit ratio, cost savings, and productivity gains, because the benefit of the new methodology, if it were great enough, would mitigate any impact that the initial cost would have. Other management factors suggested —availability of tools and experience base—would be appropriate to evaluate the maturity of the methodology. Considerations in this area included the availability of tools as opposed to just the cost, and the experience base of the methodology. Though requested, no suggestions for technical issues were submitted.

Based on the research and the survey results, several changes were made to the list of proposed evaluation considerations. Similar categories, like *Dynamic System Structure* and *Scalability*, were combined to form a single category, as were

Organizational Business Practices, *Compliance with Standards*, and *Traceability of Change*; and the *Cost of Acquiring the Methodology* and *Cost of Acquiring Support Tools*. *Methodology Maturity* was added to the list in order to capture that aspect in the decision. The final list of issues is:

- Management Issues
  - Cost of Acquiring Methodology and Tools
  - Organizational Business Practices
  - Availability of Reusable Components
  - Methodology Maturity
- Project Requirements
  - Legacy System Integration
  - Distribution
  - Environment
  - Dynamic Structure and Scalability
  - Agility and Robustness
  - Interaction

Table 1. Partition weightings

Management Issues	Technical Issues	Number of Responses
10%	90%	2
25%	75%	1
30%	70%	2
33%	66%	1
35%	65%	3
40%	60%	3
45%	55%	2
50%	50%	3
60%	40%	2
75%	25%	3
No Partition		11

4 Application of Criteria

Our research included the development of a decision-making process built upon a decision analysis framework [12]. This section describes how the criteria specified above are incorporated into the selected strategic decision-making technique.

The strategic decision-making technique selected for the problem of methodology selection is Multiobjective Decision Analysis [9]. Multiobjective Decision Analysis was selected as the underlying framework because (1) of its ability to handle multiple criteria, (2) it is based on a mathematical framework, (3) it is a flexible technique, and (4) it is a mature technique.

4.1 The Decision Analysis Tool

The first step in decision-making based on the Multiobjective Decision Analysis is the development of a value hierarchy. A *value hierarchy* is tree-like structure used for capturing evaluation considerations, objectives, and evaluation measures relevant to the decision. *Evaluation considerations* are criteria that need to be taken into account when evaluating alternatives. An o*bjective* is the preferred movement with respect to an evaluation consideration. An *evaluation measure* is a scale for measuring the degree of attainment of an objective.

For the methodology selection problem, the issues, described in Section 4, map directly to evaluation considerations in the decision problem's value hierarchy shown in Fig. 2. The objectives of the evaluation considerations, with the exception of *Cost of Acquiring Methodology and Tools*, are to maximize the rating of the methodology's ability to represent the issues. For *Cost of Acquiring Methodology and Tools*, the objective is to minimize the real dollar cost involved with acquiring the methodology and supporting tools.

In order to measure the evaluation considerations, a set of questions has been developed for each. Like the questions developed for the selection of a methodology for developing real-time systems, the questions are designed to measure the methodology's ability to represent the relevant issues [17]. Unlike the system of questions in [17], the decision-maker is asked to rate each question on a scale of zero to four. In order to capture the information, a series of worksheets have been created in [12] that collect the data, as well as provide the decision-maker with guidelines for rating each question. The purpose of the guidelines is to provide a standard for decision-makers to use while evaluating a set of subjective questions.

The Multiobjective Decision Analysis technique provides the decision-maker with a normalized score representing the fitness of an alternative with regard to the problem. This score, called the multiobjective fitness value is the additive combination of the product of the weight, *w*, and rating for each evaluation consideration, *v*. Equation 1 is the multiobjective fitness function for the decision analysis tool.

$$V(X) = \sum_{\forall i} w_i v_i(X_i) \qquad \textbf{(1)}$$

$$\text{where } i = \text{cost, bus, mat, reuse, ent, dis, env, ar, dss, int}$$

Weights are used to capture the level of importance the decision-maker places on a particular evaluation consideration. The weights make this technique flexible. The weights of evaluation considerations that are not important to the decision can be set to zero, effectively taking the evaluation consideration out of the decision.

4.2 Application of Decision Analysis Tool

The process of making the decision is captured in a decision analysis tool. This tool is the encapsulation of several data gathering steps and algebraic calculations. The process, itself, is defined by four steps:

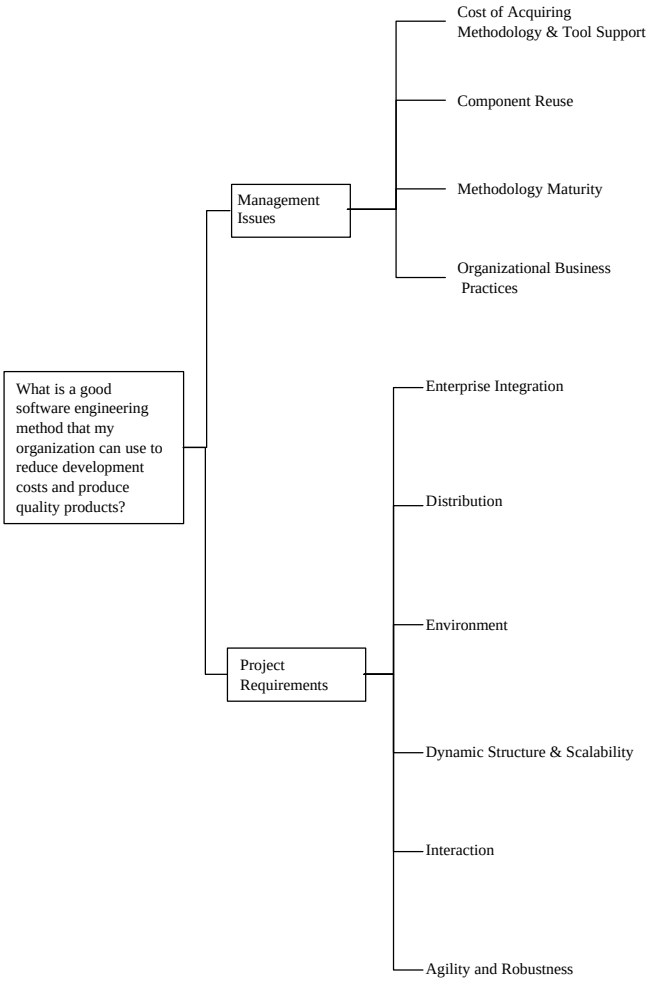

Fig. 2. Methodology selection value hierarchy

1. Weight the Evaluation Considerations
2. Rate the Relevant Evaluation Considerations
3. Calculate the Multiobjective Fitness Value
4. Determine the Best Alternative

Weighting the Evaluation Considerations involves determining which of the evaluation considerations are important to the particular software requirements problem that the decision-maker is trying to select a methodology. After determining the relevant considerations, the decision-maker determines a raw weighting for each consideration based on the relative importance each consideration has with regard to

the least important evaluation consideration. The raw weights are then normalized for use in the decision analysis.

For example, one of the case studies evaluated in [12] was based on the system requirements for a content search system [13]. The content search system is a distributed software application in which the users of the system are able to search data files throughout the users' network for key words or phrases. Based on the evaluation consideration in Fig. 2, each of the categories and the analysis decision made as to whether or not the consideration is relevant is shown below.

- *Cost of Acquiring Methodology and Support Tools* – Relevant. The approach to this problem is that the software engineer is part of an organization that is looking to adopt the methodology and supporting tools.
- *Organizational Business Practices* – Relevant. Although the software engineer is the only employee in the fledgling department, the engineer does have the responsibility of providing project updates to other interested parties outside of the department.
- *Methodology Maturity* – Relevant. The decision to change to a new methodology will require some degree of evidence that it will produce quality software.
- *Integration of Reusable Components* – Irrelevant. A library of reusable components is not available to the software engineer.
- *Legacy System Integration* – Irrelevant. The system is not required to incorporate any existing software systems.
- *Distribution* – Relevant. The users of the system will require access from different nodes on the network. Likewise, the data that the users will require is stored on many hard drives throughout the network.
- *Environment* – Relevant. The environment of the network is a mixture of Sun Workstations running Solaris OS and Personal Computers running Windows NT.
- *Agility and Robustness* – Relevant. The users of the system will expect predictability and reliability.
- *Dynamic Structure and Scalability* – Relevant. The organization is growing and as new employees are hired, the hardware systems they are given will need to be linked to the software system for access and data storage.
- *Interaction* – Relevant. The system must provide an interface for the user to submit requests.

After determining the relevance of each evaluation consideration, the decision-maker specifies weights for each. The raw weight is based on the level of importance each evaluation consideration has relative to the least important consideration. After the raw, or relative, weighting is complete, the normalized weights can be calculated. For this particular case study, the results of the relative weighting and normalization are shown in Table 2, the details for the calculation can be found in [12].

Next, the decision-maker *Rates the Relevant Evaluation Considerations* for each of the methodologies being considered. For each of the evaluation considerations considered relevant in the Step 1, the decision-maker rates the consideration by answering the respective set of questions developed during the research with respect to each alternative [12].

Table 2. Content search weighting summary

Rank	Evaluation Consideration	Relative Weight	Normalized Weight
1	Cost	1	0.172
1	Dis	1	0.172
1	Env	1	0.172
1	Int	2	0.172
2	AR	1	0.086
2	DSS	1.25	0.086
3	Mat	1	0.069
3	Org		0.069

This research evaluated an object-oriented software engineering methodology developed by Booch [1] and an agent-oriented software engineering methodology, called Multiagent System Engineering (MaSE), developed at the Air Force Institute of Technology [5] as alternatives for developing solutions to the content search problem. The documentation of the ratings can be found in [12]. For each relevant evaluation consideration, a single-dimensional value function gives the rating based on the input from the user. Table 3 summarizes the ratings of each evaluation consideration for the respective methodologies.

Table 3. Content search rating summary

Evaluation Consideration	SDVF Fitness – MaSE	SDVF Fitness – Booch
Cost	0.937	0.591
Dis	0.750	0.500
Env	0.833	0.833
Int	0.500	0.833
AR	0.417	0.250
DSS	0.750	0.625
Mat	0.333	1.000
Org	0.679	0.714

After rating each set of questions, the decision-maker has the last information needed to *Calculate the Multiobjective Fitness Values*. Using Equation 1, the weights and ratings are combined to form a single fitness value for each alternative. In the case of the evaluation considerations that were determined to be irrelevant, the term can be dropped or a zero can be entered. An example of the calculation is shown below for the MaSE alternative.

$$\begin{aligned}\mathbf{V_{MaSE}(X)} &= w_{cost}v_{cost}(x_{cost}) + w_{org}v_{org}(x_{org}) + w_{mat}v_{mat}(x_{mat}) + w_{dis}v_{dis}(x_{dis}) \\ &\quad + w_{env}v_{env}(x_{env}) + w_{ar}v_{ar}(x_{ar}) + w_{dss}v_{dss}(x_{dss}) + w_{int}v_{int}(x_{int}) \\ &= 0.172\ vcost(1690) + 0.069\ vorg(19) + 0.069\ vmat(4) + 0.172\ vdis(9) \\ &\quad + 0.172\ venv(10) + 0.086\ var(5) + 0.086\ vdss(6) + 0.172\ vint(6) \\ &= 0.161 + 0.047 + 0.023 + 0.129 + 0.143 + 0.036 + 0.065 + 0.086 \\ &= \mathbf{0.689}\end{aligned}$$

The summary of multiobjective fitness values (MFV) is shown below in Table 4.

Table 4. Content search MFV summary

Case Study	MaSE MFV	Booch MFV
Content Search System	0.689	0.668

With the multiobjective fitness values for each alternative, the decision-maker has a quantified value to base the decision. For this example, the decision analysis tool recommends MaSE over Booch. In cases where the results are close, there are a number of techniques for evaluating the sensitivity of the decision based on the weights assigned in step 1.

4.3 Sensitivity Analysis

The two factors that determine the value of the multiobjective are the weights assigned to each evaluation consideration and the score the alternatives receive for each evaluation consideration. Though the ratings of each of the focus points are subjective, each rating is based on the assessor's experience and we assume it is accurate. However, because the weights are defined strictly on a perceived importance of one evaluation consideration over another, we are wary of their accuracy and subject them to sensitivity analysis. To give the assessor a feeling for the definitiveness of the decision, we can perform sensitivity analyses on the weightings of each evaluation consideration. We focus this analysis on the areas where a slight change in an evaluation consideration's weight could significantly change the overall fitness score.

Using the Data Analyzer tool that we developed to work with the output of the decision analysis tool, a full sensitivity analysis can be performed on the weights of all of evaluation considerations. The analysis focuses on the most sensitive of the considerations with regard to the original normalized weight by evaluating the fitness of each methodology over a range of weights for a particular methodology. For our case study, we calculated the entire range of possible weight, from 0 to 1. To ensure that the total normalized weight remains 1, the other considerations (those not currently being analyzed) are adjusted to be proportional to the total weight minus the weight of the consideration being analyzed. We consider the final decision *sensitive* if a small change to the weight – a change of 5% to 7% [9] – produces a change in the final decision. We consider a recommendation is *definitive* when the percentage of sensitive considerations is less than 33% [9].

Detecting sensitivity relies on the identification of critical points. A *critical point* is where the weight for the particular consideration changes the decision analysis tool's preference. The sensitivity analysis chart for *Methodology Maturity* of our case study is shown in Fig. 3. The analysis chart is annotated to highlight the critical points. For example, this figure has three critical points. The first is when the weight for *Methodology Maturity* is 0.098. When the weight is within the range 0 to 0.098, the MaSE methodology has the highest multiobjective fitness value. The second critical point, at 0.382, is the weight that the Yourdin methodology rates begins to rate higher than MaSE, however, it is still less than the Booch methodology. The third

critical point, 1.000, is where the Booch and Yourdin methodologies intersect. However, by setting the weight of *Methodology Maturity* to 1.000, it is the only factor being taken into account.

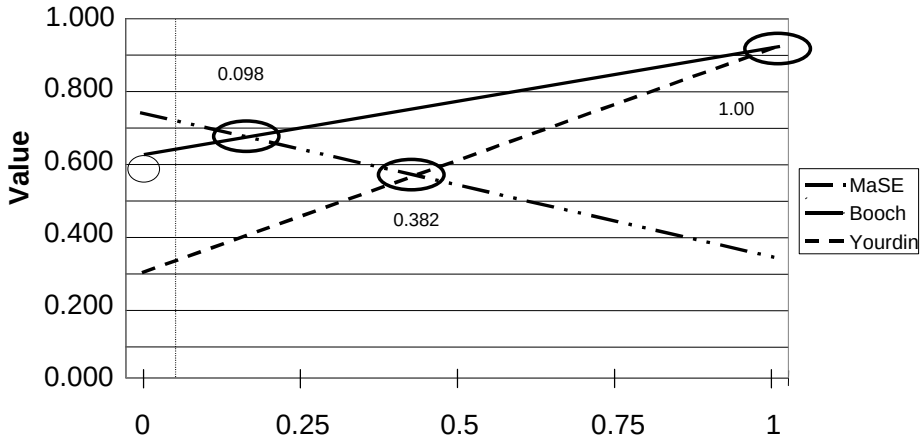

Fig. 3. Methodology maturity sensitivity analysis chart

The critical points and original weights for our case study are shown in Table 5. For the considerations where there is no critical point, "-" is entered as the critical point. Additionally, the amount of the change in weight needed to alter the final decision is noted with changes less than and equal to 7% highlighted in bold font.

Table 5. Case study weights and critical point summary

Evaluation Consideration	MaSE – Booch Critical Point	MaSE – Yourdin Critical Point	Booch – Yourdin Critical Point	Original Weight	Change (+/-)
Cost	0.117	-	-	0.172	**0.056**
Org	0.419	1.000	-	0.069	0.350
Dis	0.094	-	-	0.172	0.078
Env	1.000	-	-	0.172	0.828
AR	-	-	-	0.086	-
DSS	-	-	-	0.086	-
Int	0.223	1.000	-	0.172	**0.051**
Mat	0.098	0.382	1.000	0.069	**0.029**

Our case study was sensitive to three criteria out of six making the answer produced non-definitive. While our decision analysis recommended MaSE, our the sensitivity analysis indicates that if the user's weighting preferences were slightly different, the Booch methodology could easily win. In all likelihood, either methodology would satisfy the user's needs.

4.4 Validation of Decisions

The decision analysis tool was demonstrated on a number of example software requirements in [12]. The challenge with validating the decision the tool returns is that the decision is being made based on subjective criteria. As an example, the software requirement for the content search was developed via the two rated methodologies—MaSE and Booch.

During the development process, a set of metrics was collected. The metrics collected focused on the productivity of the developer. They included labor hours spent developing the analysis and design models and the implementation, the size of the programs measured in lines of code and number of components, and the complexity of the developed code. The time spent analyzing and designing the systems were similar, but more time was spent on the object-oriented implementation. The size of the agent-oriented code was roughly twice as large as the object-oriented code. The data collected is shown in Table 6.

Table 6. Content search development metrics

Metric	MaSE Approach	Booch Approach
Modeling Effort – Analysis	4.83 labor hours	4.53 labor hours
Modeling Effort – Design	2.17 labor hours	4.08 labor hours
Modeling Effort – Implementation	8.17 labor hours	11.75 labor hours
Size – SLOC	1252	638
Size – Classes	20	11
Cyclomatic Complexity	74	6
Size/Effort Ratio	153.2 SLOC/labor hour	54.3 SLOC/labor hour

In addition to collecting the metrics, a questionnaire was distributed to a class of software engineering graduate students. The purpose of the questionnaire was to determine whether the details of the system's requirements were identifiable within the analysis and design models of the respective methodologies. The students reviewing the agent-oriented analysis and design models scored higher than those reviewing the object-oriented did. This corresponds with the decision analysis tool's determination that the agent-oriented methodology was more appropriate the requirement [12].

The first set of questions the respondents answered was to their familiarity with methodologies they were evaluating. Eight of the nine students reviewing the Booch models indicated that they were familiar with the methodology, and five of those indicated that they had developed systems using the methodology in the past. Only six of ten students indicated that they were familiar with the MaSE methodology, and of those, only five students had actually used the methodology for system development. These results were expected since MaSE is a recently defined agent-oriented methodology while the Booch methodology is much more mature. Asked to identify other methodologies with which they are familiar, the students indicated object-oriented techniques, functional decomposition techniques, and ad hoc methods for developing software.

The next question was a general question about the respondents' confidence that they understood the models. Each was asked to rate his confidence on a scale of zero to four, with four indicating the greatest confidence in understanding the system. On the average, the understanding rating for the students evaluating the MaSE methodology was 3.2. The rating was 3.125 for the students evaluating the Booch methodology. Seven of the eight students reviewing the Booch methodology were able to identify the correct statement of description for the system. Only two of the ten students reviewing the MaSE methodology were able to select the correct statement; the other eight students selected the "nearly" correct answer.

The next set of questions looked at a number of details in the models, including the identification of legacy systems, reusable components, the network environment, and interface issues. The students reviewing the Booch methodology were divided equally with regard to identifying a legacy system. Because there was not a legacy system incorporated in this system and the responses as to what the legacy system could possibly be, the naming convention was likely the reason for the misidentification. Only one student misidentified the legacy system in the set of MaSE models.

Determining the network environment was the intention of several questions. Identifying the configuration of the network the system was being designed for is important information that needs to be communicated to the developers. These questions measured the respondents' ability to discover this environment information. The group evaluating the MaSE example was able to more completely identify the hardware and software system components in the models. With regard to the user interface, both groups were able to identify the input and output of the systems as well as the options.

Based on the scoring included next to each question on the questionnaire in [12], the average scores are 25.5 for the MaSE group and 25.4 for the Booch group. Furthermore, by considering the results for the students who were familiar and experienced with the respective methodology, the average score for the MaSE group was 27.2 and for the Booch group was 25.1.

The student responses pointed out positives and negatives associated with each approach. However, the results of this experiment are consistent with the results produced by our decision analysis tool.

5 Conclusions

The reasons for software engineering methodologies are clear: develop a high quality software product at the least cost. When faced with the challenge of creating one of these high quality/low cost products, it is necessary to use the methodology that best fits the problem. The challenge is, "how do you decide what the best method is?"

The challenge becomes even greater as methodologies are developed that specifically address new technologies, such as the development of multiagent systems. Agent-oriented software engineering provides a different way of looking at the same problem by raising the level of abstraction. The solution for this is to be able to classify different software engineering methodologies quantitatively based on the software requirement at hand.

The challenge in this is developing a set of criteria that represents the problem space. To generalize this problem space, we developed a set of criteria from current software engineering literature. To ensure that others agree with our criteria, we invited various members of the software engineering community to participate in a survey. Based on the results of this survey, we adjusted the criteria to include additional factors that we missed as well as to remove criteria the community did not find important.

The method provides the user with the ability to determine the best methodology for a particular problem. There is still an outstanding question of when to use multiagent systems. The challenge with this is that there does not exist a large body of evidence to support the hypotheses that multiagent systems are superior to traditional systems. Because there is currently so much research focused on developing new methodologies, more multiagent systems will inevitably be created, which, in turn, will create a larger body of data to compare with traditional systems.

6 Acknowledgements

This work was performed while both authors were at the Air Force Institute of Technology and was supported by the Air Force Office of Scientific Research. The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the US Government.

References

[1] Booch, G.: Object-Oriented Analysis and Design with Applications. The Benjamin/Cummings Publishing Company, Redwood City, CA (1994)

[2] Brauer, W., Nickles, M., Robatsos, M., Weiss, G., Lorentzen, K.: Expectation-Oriented Analysis and Design. In this volume (2001)

[3] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Modeling Early Requirements in Tropos; a Transformation Based Approach. In this volume (2001)

[4] Caire, G., Garigo, F. Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., Massonet, P.: Agent Oriented Analysis Using MESSAGE/UML. In this volume (2001)

[5] DeLoach, S. A., Wood, M., Sparkman, C.: Multiagent Systems Engineering. To appear in the Intl. J. on Software Engineering and Knowledge Engineering (2001)

[6] Firth, R., et al.: A Classification Scheme for Software Development Methods. Software Engineering Institute Technical Report 87-TR-41. Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA (1987)

[7] Iglesias, C.A., Garijo, M., Gonzalez, J.C.: A Survey of Agent-Oriented Methodologies in Intelligent Agents V – Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98), Lecture Notes in Artificial Intelligence, Vol. 1555. Springer-Verlag, Berlin Heidelberg New York (1998)

[8] Jennings, N.R., and Wooldridge, M.J.: Agent-Oriented Software Engineering. To appear in Bradshaw, J. (ed.): Handbook of Agent Technology. AAI/MIT Press (2001)

[9] Kirkwood, C. W.: Strategic Decision Making: Multiobjective Decision Analysis with Spreadsheets. Wadsworth Publishing, Belmont, California (1997)
[10] Lesser, V.R.: Cooperative Multiagent Systems: A Personal View of the State of the Art. IEEE Trans. on Knowledge and Data Engineering. **11** (1) (1999) 133-142
[11] MESSAGE: Methodology for Engineering Systems of Software Agents – Initial Methodology. EURESCOM Participants in Project P907-GI (2000)
[12] O'Malley, S.A.: Selecting a Software Engineering Methodology Using Multiobjective Decision Analysis, AFIT/GCS/ENG/01M-08. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH (2001)
[13] O'Malley, S.A., Self A., and DeLoach, S.A.: Comparing Performance of Static versus Mobile Multiagent Systems. Proceedings of the National Aerospace and Electronics Conference. IEEE (2000) 282-289
[14] Omicini, A.: SODA: Societies and Infrastructures in the Analysis and Design of Agent-Based Systems. In Ciancarini, P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering: First International Workshop, AOSE 2000. Lecture Notes in Artificial Intelligence, Vol. 1957. Springer-Verlag, Berlin Heidelberg (2001) 185-194
[15] Rana, O.: A Modelling Approach for Agent Based Systems Design. In Ciancarini, P., Wooldridge, M. (eds.): Agent-Oriented Software Engineering: First International Workshop, AOSE 2000. Lecture Notes in Artificial Intelligence, Vol. 1957. Springer-Verlag, Berlin Heidelberg (2001) 195-206
[16] Shen, W. and Norrie, D.: Agent-Based Systems for Intelligent Manufacturing: A State-of the-Art Survey. Intl. J. Knowledge and Information Systems. 1 (2) (1999) 129-156
[17] Wood, B., Pethia, R., Gold, L.R., and Firth, R.: A Guide to the Assessment of Software Development Methods. Software Engineering Institute Technical Report 88-TR-8. Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA (1988)
[18] Wooldridge, M.J.: Intelligent Agents. In Gerhard Weiss (ed.): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge, Massachusetts (1999)
[19] Zhu, H.: A Formal Specification Language for MAS Engineering. In this volume (2001)