Prioritisation of System Changes using Cost-Benefit and Risk Assessments

D. Greer and D.W. Bustard
School of Information & Software Engineering,
University of Ulster, Coleraine, UK
d.greer@ulst.ac.uk

T. Sunazuka,
SPEED Technology Laboratory,
NEC Corporation,
Tokyo, Japan

Abstract

This paper proposes an approach to the prioritisation of system changes that takes account of the relative costs and benefits of those changes and the risks that they reduce or introduce. This is part of the SERUM methodology (Software Engineering Risk: Understanding and Management), which is being developed to help identify effective ways of using risk analysis and control in software production. SERUM introduces risk management at the initial business analysis stage of requirements investigation, and assumes an evolutionary approach to software delivery. Prioritisation is determined from five factors: benefits, costs and risk exposure in the current system, target system, and development process. The relative importance of these factors is adjustable. Results from a case study at NEC illustrate the prioritisation process and a supporting software tool is also described.

1. Introduction.

Lehman's well known *Law of Continuing Change* [12] observes that a program used in a real-world environment must change or become progressively less useful. Put another way, it means that change is inevitable for all commercial software systems. This inevitability arises from the fact that a system's environment is also constantly changing. Planning which changes should take place is rarely straightforward. Typically, for any application, there will be a backlog of potential changes. It is impractical to implement all of these simultaneously because of the cost involved, staff limitations and the overall time required to make the changes. Thus, some form of prioritisation is needed to facilitate change selection for each new system version.

One (traditional) approach is to first tackle those changes that provide the best cost-benefit ratio. A complication, however, is that there is *risk* associated with any change. Users, for example, often refuse available upgrades because of the problems they may introduce, preferring instead to persist with 'proven' software. Such *technical risk* [4], that is risk in the operation of software, is an important consideration in planning which software changes to implement first. Some changes may reduce technical risk while others increase it.

Development risk, that is risk associated with the process of implementing a change, is another factor that can affect priority. This is particularly true when there is uncertainty over the technical feasibility of a change, but is also influenced by the many risks associated with managing the development process.

The challenge then is to find some means of prioritising changes to take account of these factors in an appropriate way. In this paper, SERUM (Software Engineering Risk: Understanding and Management) [7], is offered as one possible approach. It makes use of Soft Systems Methodology (SSM) [3, 15] to provide a business analysis for the system being investigated. SSM is essentially a goal-oriented problem solving technique. Its application produces a model of an 'ideal world' identifying activities that need to be performed to meet the agreed goals of the system. From this model a set of recommendations for change is derived. In some cases the changes may be purely organisational but more commonly they involve technological change to introduce or enhance computing support within the organisation. These change recommendations effectively define the differences between the current position and the ideal model.

An application of SSM will often identify substantial changes to an existing way of working. Again such changes cannot all be implemented at the same time. Cost is a key factor but so also is the risk associated with the disruption to the organisation concerned. Hence, an evolutionary development model has been adopted for planning these changes [2, 3]. This offers substantial risk-reducing benefits [3].

In software evolutionary development [6], parts of a system are implemented and delivered in phases. Each delivery is a complete system that is of value to the client. The delivered system is evaluated by the client and the results fed back to the developers who then take that information into account when implementing subsequent phases. SERUM extends this approach to include organisational change.

The next section reviews current approaches to prioritising requirements that could be used for

scheduling system changes. Possible prioritisation criteria are then identified, together with a discussion of their assessment means. The section that follows this describes the SERUM methodology and how it uses these measurements to develop a delivery plan for system changes. This is illustrated with an example based on a research study at NEC Corporation in Tokyo.

2. Current Approaches to Prioritisation.

Current approaches to prioritising requirements make use one of two assessment techniques:

(i) Relative assessment: ordering requirements by comparing groups of them with each other against a defined set of criteria; and

(ii) Absolute assessment: estimating requirements in absolute terms, judged against defined criteria, with the assessment results defining the priority order.

The most popular approaches to the prioritisation of requirements through relative assessment are based on the *Analytical Hierarchy Process (AHP)* [13]. With AHP, the relative importance of the assessment criteria is first determined through their pair-wise comparison. For example, if criteria for assessing system changes are taken to be *cost-benefit ratio*, *impact on system quality* and *risk-reduction*, it might be decided that *cost-benefit ratio* is twice as important as *impact on system quality* and that it in turn is twice as important as *risk-reduction*. For consistency, the *cost-benefit ratio* should emerge as four times as important as *risk-reduction* (Table 1).

Table 1: Priorities of Criteria (AHP)

	Cost-Benefit	Impact on Quality	Risk Reduction
Cost-Benefit	1	2	4
Impact on Quality	0.5	1	2
Risk Reduction	0.25	0.5	1

Using these scores weighting for each criterion is derived. It can be calculated by dividing the reciprocal of the entry by the sum of the reciprocals of all the entries in the row. Thus using the first row (the other rows are used to check consistency), the following figures emerge.

Cost-Benefit: 1 / (1 + 0.5 + 0.25) = 0.57
Quality Impact: 0.5 / (1 + 0.5 + 0.25) = 0.29
Risk Reduction: 0.25 / (1 + 0.5 + 0.25) = 0.14

Table 2: Preferences (AHP)

Quality Impact	Desk Checking	Fagan Inspection
Desk Checking	1	1/2
Fagan Inspection	2	1

A similar approach is then used to assess each candidate requirement in relation to the chosen criteria. So, for example, a pair-wise comparison of desk-checking of code against Fagan inspections [5] might deduce that they were similar in relation to cost-benefit ratio, but that Fagan inspections had a higher impact on quality and risk-reduction (Table 2). Again, using the first row, the preference scores are:

Desk Checking: 1 / (1 + 2)= 0.33
Fagan Inspection: 2 / (1 + 2) = 0.67

The same scoring mechanism is used for all comparisons, so that each requirement obtains a preference score with respect to each decision criterion. The overall rating for a requirement is obtained by summing the preference scores and multiplying by the weighting for that criterion.

The disadvantage of AHP is that for a substantial number of requirements, a very large number of pair-wise comparisons are needed, which can quickly become unworkable. For example with twenty requirements, a total of one hundred and ninety (n*(n - 1)/2) pair-wise comparisons must be performed for each criterion. Even if techniques are used to reduce the number of comparisons [10], the effort is still considerable and usually provides less consistent results [11]. In the AHP example above, the number of comparison can be reduced to (n-1) by using just the first row of the preferences table (Table 2), but this means sacrificing the consistency check. Other, less widely accepted prioritisation approaches include using a bubblesort or building a binary tree. However, both of these approaches require abstraction across criteria rather than comparison against individual criteria and so, perhaps, are less reliable.

Requirements can also be assessed directly against criteria rather than relative to each other. The use of such absolute measurement has a number of distinct advantages over the relative comparison approach. One is that fewer assessments are needed – just one for each requirement against each criterion. Another is that it avoids the need to compare requirements with each other, which can be difficult when they are often unrelated.

With both approaches, the number of criteria used for assessment will dictate the overall effort required and so should be kept as small as possible to achieve the desired ordering. The next section considers which criteria might be suitable.

3. Prioritisation - Candidate Criteria

The evolutionary delivery process promoted by Gilb [6] involves the prioritisation of system changes in preparation for a phased delivery. Gilb advocates the ‘user-value to development cost ratio’ to determine the order of delivery. In many, perhaps most cases, user-value is equivalent to the benefit:cost ratio arising from a system change.

In general, we agree with the concept of ‘user value’

as described by Gilb, but seek to define what exactly is meant by the term. Our hypothesis is that user-value for most system changes means high benefit, low cost, a significant technical risk reduction, and low development risk.

Costs and *benefits* are traditionally considered appropriate measures for any business investment. Almost certainly a system change will have an associated cost and this can be measured in terms of monetary outlay. Similarly, benefits can be assessed in terms of extra income generated.

Each system design can be assessed in terms of the technical risks that it embodies. Risks in this context represent a potential for error and so for the benefit of users should be kept as low as possible. System changes, therefore, should also aim, if feasible, to resolve, or at least reduce such technical risks.

Technical risk arises when there is the possibility of a system causing some loss during its operation. To measure these risks the probability of the unwanted event occurring and the impact of the event need to be assessed. Boehm defines *Risk Exposure* (RE) [1] as the product of the probability of a risk occurring and the impact of such an event.

*RE = prob (risk) * impact associated with risk*

The choice of a unit for impact is a difficult one. It may be given in financial terms, but other units such as 'down time' may be more appropriate in some circumstances.

System changes may also introduce new technical risks. Measuring technical risk in a proposed system is similar to measuring it in the current system. Again a risk exposure calculation may be employed combining the probability of an unwanted event and the impact should it occur.

Finally, there are *development risks* to consider. These include project risks in budget and schedule and can strongly influence the decision on whether a proposed change is viable. Development risk can be measured in a similar way to technical risk.

The priority of change can thus be expressed as a function of five variables.

- *risk exposure in the current system,*
- *risk exposure in the proposed system,*
- *risk exposure in the implementation of a change*
- *cost of a defined change*
- *benefit of a defined change*

The emphasis given to each factor will vary with circumstances to some extent. Benefit is the over-riding concern but this must be tempered by cost and risk considerations. Clearly, an important change is less attractive if it is expensive to implement or there is significant uncertainty over the likelihood of success. The relative importance of cost and risk must be determined by the decision-makers, however, and the objectives in providing support are to determine how contributing factors should be assessed and in what form they should be presented to facilitate the decision-making process. The SERUM methodology was developed to meet this need.

4. SERUM Methodology.

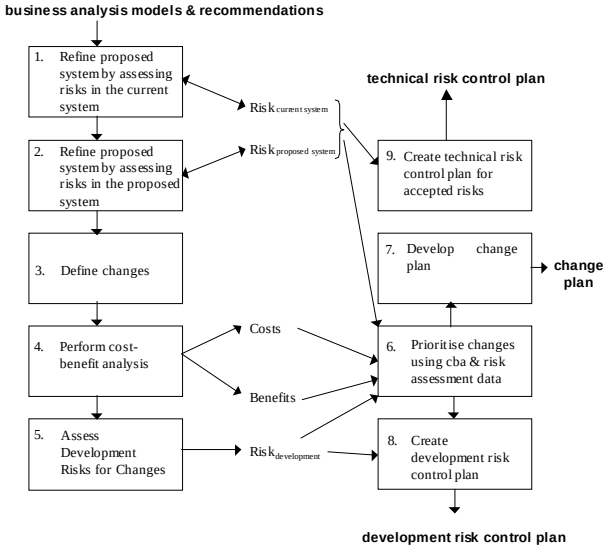

Figure 1: Overview of SERUM

The SERUM Methodology, summarised in Figure 1, integrates risk management into an overall technological change process for an organisation. As such it is particularly applicable to the planning of software version releases.

SERUM takes as input, business analysis models and change recommendations developed using Soft Systems Methodology (SSM). These models identify the objectives of an organisation and the activities necessary to achieve those objectives. As such they describe an 'ideal world'. An analysis of how this description differs from the current way of working leads to a basic set of recommendations for change. SERUM is concerned with (i) refining the models and recommendations, by taking account of risk; (ii) the production of an evolutionary change plan; (iii) the production of a technical risk control plan; and (iv) the production of a development risk control plan. For a fuller description of how SSM is integrated with SERUM and a discussion of the risk reducing benefits gained from its use, see [7].

The following explanation of the SERUM methodology is illustrated with results from a recent study undertaken at NEC [9]. The study included an evaluation of SERUM in a project for planning future releases of a Network Management System (NMS) at NEC. The NMS is used by large telecommunications and internet service providers for centralised

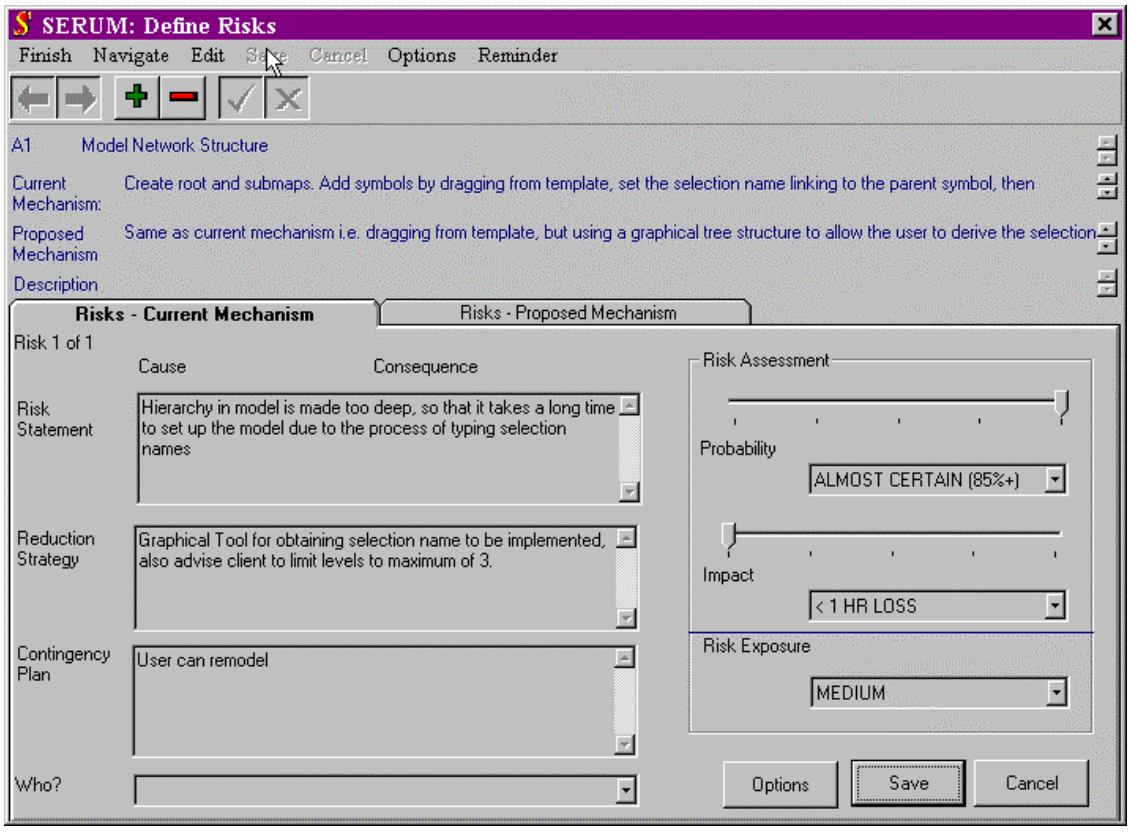

Figure 2: SERUM tool support for the definition of technical risks

management of networks distributed across cities, countries or even continents. Physically, the NMS consists of at least one server and several workstations connected to the managed network. The NMS software allows a network to be modelled in terms of standard responsibilities such as network performance, fault handling, configuration management, accounting and security. The managed network is modelled graphically.

The case study was carried out on-site at NEC's Overseas Transmission Division. Data was gathered from four senior engineers working on the project. The risk and cost-benefit assessments were carried out using the tool support with SERUM and in consultation with the appropriate project expert. Most of the feedback for the case study was obtained from the senior engineer responsible for planning and overseeing future releases of the Network Management System.

In the first two stages of SERUM, the SSM models and recommendations are refined through an analysis of the technical risks involved. The objective is to identify revisions that will reduce or eliminate such risks as far as far as is practical. Technical risks are related to possible losses due to the way that activities are performed. Soft Systems Methodology (SSM) models identify those activities.

Risk exposure can be calculated numerically but it seems more appropriate to make clear that the assessment is approximate and so use symbolic values instead. Table 3, for example, shows the approach that was taken in the NMS study.

Table 3: Risk Exposure for Technical Risk

Probability ***Impact***	almost certain (85%+)	very likely (60-84%)	likely (40-59%)	unlikely (20-39%)	very unlikely (< 20%)
> 2-day loss	Very high	Very high	High	Medium	Medium
2-day loss	Very High	High	High	Medium	Medium
1-day loss	High	High	Medium	Low	Low
1-hour loss	Medium	Medium	Low	Low	Very low
<1-hour loss	Medium	Medium	Low	Very low	Very low

The probability or likelihood of occurrence is measured on a five-point scale from ‘almost certain’ to ‘very unlikely’, while the loss impact is similarly rated over five values. It is expressed in terms of the loss of service, from ‘greater than two days’ to ‘less than one hour’. The elements of the table define the risk exposure, which ranges from ‘very high’ to ‘very low’. This is adequate to identify high-risk areas where further investigation is desirable.

SSM models may contain over a hundred activities

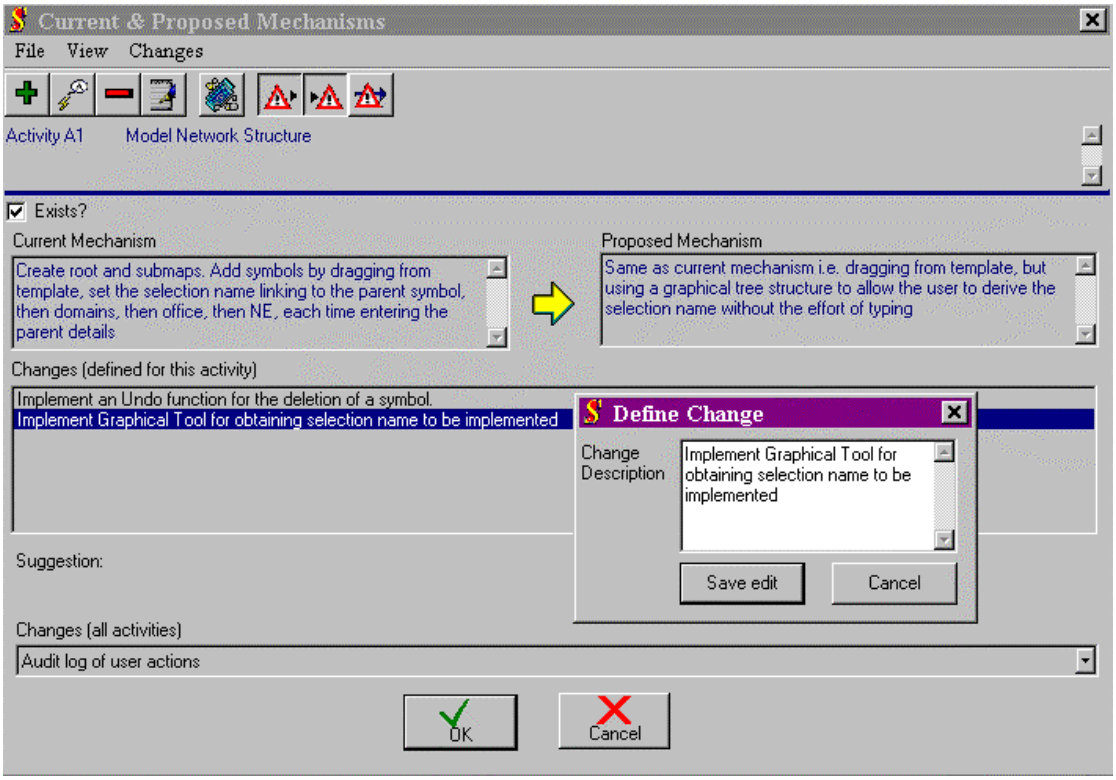

Figure 3: SERUM tool support for definition of system changes

and so some form of tool support is desirable to help define and adjust risk assessments against those activities. The SERUM tool allows risk measurement scales to be defined for each application and then enables related assessments to be entered. Figure 2 shows the window through which risks are defined. Risks are specified for each activity, in this case the activity is "A1: model network structure". Convenient access to both current and proposed mechanisms is provided for comparison purposes.

In the example in Figure 2, the risk identified is that a user may mistype the hierarchical path to a network element. The proposed improvement is to use a tool to support selection from a graphical representation of the network hierarchy to eliminate the possibility of error. In assessing risk in the current and proposed mechanisms there is an opportunity to suggest mitigation strategies for the risk and to plan contingencies, should the associated problems arise. This will contribute to an overall risk control plan developed at a later stage.

In stage three, possible system changes are defined. Each activity may have several changes associated with it and some changes may be relevant to several activities. The description of each change and its linkage to activities is developed though the window shown in Figure 3.

In stage four the cost and benefit of each change is examined. At this level of analysis it is again adequate to use a symbolic assessment rather than develop detailed financial estimates. Costs and benefits are often difficult to quantify precisely so an approximation approach will save effort while still providing sufficient information to support the prioritisation of change. Full costings are generated later for those changes that proceed to implementation.

Table 4: Matrix for obtaining the cost-benefit assessment of a proposed change

Cost *Benefit*	2-3 months	1 month	2-3 weeks	1 week	< 1 week
Very High	OK	OK	Good	Very Good	Best
High	OK	OK	Good	Good	Very Good
Medium	Poor	OK	OK	Good	Very Good
Low	Very Poor	Poor	Poor	OK	Good
Very Low	Worst	Very Poor	Very Poor	Poor	OK

Table 4 shows the cost-benefit matrix developed for the NMS study. Benefits range from 'very high' to 'very low'. Costs are expressed in terms of the effort required to complete an implementation, ranging from '2-3 months' to '< 1 week'. These phrases are related to scores as defined by the analyst. The resulting cost-

benefit shown in the matrix in the NMS study was rated from 'worst' to 'best' across a seven-point scale.

Again the SERUM tool supports the definition of these values and the entry of cost and benefit assessments, as illustrated in Figure 4.

Figure 4: SERUM tool support for costs-benefit analysis

Development risk is considered in stage 5. This uses the same approach as technical risk. Table 5 illustrates the ratings used in the NMS case study.

Table 5: Assessing development risk exposure

Probability / *Impact*	Almost certain (85%+)	Very likely (60-84%)	Likely (40-59%)	Unlikely (20-39%)	very unlikely (< 20%)
Very High (20%)	Very High	Very high	High	Medium	Medium
High (15-20%)	Very High	High	High	Medium	Medium
Medium (10-15%)	High	High	Medium	Low	Low
Low (5-10%)	Medium	Medium	Low	Low	Very low
Very Low (< 5%)	Medium	Medium	Low	Very low	Very low

Again the impacts are expressed symbolically (as defined by the analyst) but represent a figure estimating the extra cost to development. Thus, a 'high' impact means 15% to 20% extra cost. Again, tool support for the estimation and documentation of development riks has been implemented in a similar way to technical risks.

Figure 5 shows an example from the NMS study. Here, the change defined assumes that the GUI facilities in the current implementation are sufficient to build an undo function. As there is some doubt that this is true, a risk is recorded. The right hand side of the window shows the assessment of this risk, namely that the probability is 'not likely' and the impact is 'very low'. This window also provides an opportunity to record mitigation strategies and contingencies for use in the risk control plan.

Stage 6 gathers together the benefit, cost and risk assessments from the earlier stages to facilitate prioritisation. Just as cost-benefit ratio has emerged as an effective way of combining these factors, Risk Reduction Efficacy (RRE) is here introduced as a way of combining a consideration of technical risks in the current and proposed systems. Risk reduction is desirable overall but may not always be achievable. For example, moving to a new hardware platform increases technical risk but may be essential to meet performance needs.

RRE for a given system change is defined as

$$RRE = \sum_{All\ Technical\ Risks} \left(RE \times contribution \right)$$

where

RE is risk exposure for a risk identified in the current or proposed system,

contribution is a factor for the *increase or decrease* in risk exposure due to the change, being positive or negative respectively, and

All Technical Risks refers to each instance of a risk defined in the current and proposed system.

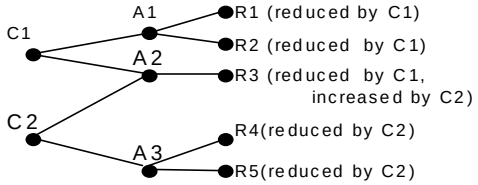

Figure 5: Relationships for changes, activities and risks

Figure 5 illustrates the type of relationship that can exist between system changes, activities in the current system, and the associated risk. In each case, risk exposure may be reduced or increased. Change C1 affects activities A1 and A2; change C2 affects A2 and A3. Activity A1 has associated risks R1 and R2; A2 has risk R3; and A3 has risks R4 and R5. To find how much a given system change reduces risk, it is necessary to assess its contribution to each individual risk exposure for each of the activities affected. For change C2, there is a contribution from R3, R4 and R5, with R4 and R5 reduced and R3 increased. RRE will then be the sum of the three products combining exposure and contribution. If, say, C2 reduces risk exposure R4 by 50% and R5 by 40% and that C2 increases the risk exposure R3 by 20%, the RRE would be (R4*0.5) +(R5*0.4) + (R3*-(0.2))

Relative Risk Reduction Efficacy (RRRE) for a system change may then be defined as follows:

$$RRRE = \frac{\sum_{All\ Technical\ Risks} (RE \times contribution)}{\sum_{All\ Current\ TechnicalRisks} RE_C}$$

The denominator is the sum of the risk exposures for the risks in the current system. Using RRRE, it is then possible to compare system changes for efficacy in risk reduction and to use this data in prioritising system changes for an evolutionary delivery plan. In Stage 7 the prioritisation is completed by defining the relative contribution from each criterion. This can be achieved using weightings as derived using the AHP technique described in the first section (Table 2). Another (simpler) approach is to order the criteria and sort on the risk assessments. In practice this approach was found to be satisfactory when accompanied with additional adjustments based on the data collated in the SERUM tool (Figure 7).

The main criterion is usually cost-benefit but where the costs are all reasonable, benefit alone may be the main indicator for priority as was the case in the NMS study. Because the assessments are made symbolically, several changes will often have the same rating so further sorts can be performed. Table 6 gives a sample of the ranked changes from the NMS study using a sort that closely matched the intuitive priority order suggested by a member of the development team.

The SERUM tool collates and displays the information previously entered on costs, benefits and risks and allows the analyst to manually adjust the order

Figure 6: SERUM tool support for sorting

previously derived from sorting (Figure 6). For example, if benefit was the primary sorting key, it is possible that there are may be changes unjustifiably high in the list. This situation may arise in the case where a change has a very high benefit but also very high costs or an unacceptable development risk exposure.

In Stage 8 a development risk plan for the change

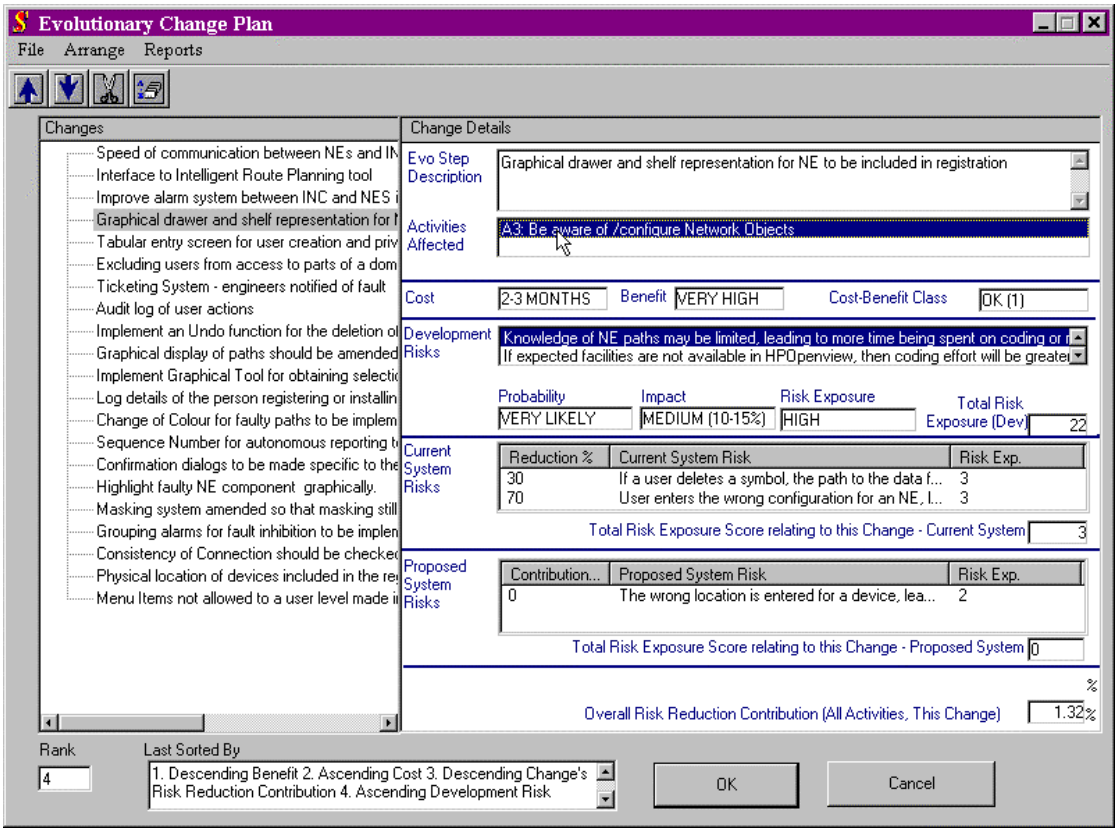

Figure 7: SERUM tool support for prioritisation of evolutionary delivery

Table 6: Sample Ranking of System Changes from SERUM

Last Sort: 1. Desc. Benefit 2. Asc. Cost 3. Desc. Change's Risk Reduction Contribution 4. Asc. Development Risk Exposure

Rank	Change Description	Cost	Benefit	Cost-Benefit	Risk Reduction Efficacy	Total Development Risk Exposure Score
1	Speed of communication between NEs and INC-100 - Improve (A6)	1 MONTH (4)	VERY HIGH (5)	OK (0.8)	0.44 %	15
2	Interface to Intelligent Route Planning tool (A22 A5)	2-3 MONTHS (5)	VERY HIGH (5)	OK (1)	2.64 %	44
etc						

process is extracted from the information supplied in earlier stages. Finally, in Stage 9, plans for handling technical risks in the current and proposed systems are similarly collated.

5. Conclusions.

This paper has demonstrated how the SERUM methodology can be used to prioritise system changes. This involved assessing the benefit of each change, its associated cost, the risk in making the change, and the extent to which the change increased or decreased risks in the current system. Each change was assessed directly against each criterion rather than using the more common approach of relative comparison. This reduced the number of assessments needed to determine a ranking. In the NMS case study employed, we have used the assessments from SERUM along with analyst expertise to prioritise system changes.

We have further indicated how the method can be enhanced using criteria weightings established using the Analytic Hierarchy Process.

The SERUM approach can be used to prioritise changes emerging from any source. Also, as illustrated in this paper, it can be used with broad approaches to change definition such as Soft Systems Methodology (SSM). It also makes several contributions to SSM itself in that (i) it offers a systematic way of developing recommendations for change; and (ii) suggests how business models can be enhanced through risk analysis.

Overall, SERUM emphasises the basic importance of risk assessment in ensuring that systems are implemented effectively and are designed to be robust.

Acknowledgements

Thanks are due to NEC Corp, particularly to Mr M. Yoshimura and Mr. Y. Higaki, of NEC's Overseas Transmission Division, for their contribution in providing data and expertise for the case study. Funding and support from The British Council and The Great Britain Sasakawa Foundation and are also acknowledged. SERUM is also contributing to the RIPPLE project (Retaining Integrity in Process Products over their Long-term Evolution) funded by EPSRC, GR/L60906, under the SEBPC (Systems Engineering for Business Process Change) Programme. RIPPLE is concerned with linking business and computing models as a basis for ensuring that computing facilities are directly supportive of a business and also remain aligned as business and computing changes occur.

References

1. Boehm, B.W., Software Risk Management: Principles and Practices, *IEEE Software*, January, pp 32-41, 1991.
2. Bustard, D.W., and He, Z., A Framework for the Revolutionary Planning and Evolutionary Implementation of a Business Process and its Computing Support, *Logistics Information Management*, 11, 6, pp. 370-374, Nov. 1998.
3. Checkland P.B. & Scholes J: *Soft Systems Methodology in Action*, Wiley, 1990
4. Chittister, C. and Haimes, Y.Y., Assessment and Management of Software Technical Risk, *IEEE Trans. On Systems, Man and Cybernetics*, 24, 2, 187-202, 1994.
5. Fagan, M.E., Design and Code Inspections to Reduce Errors in Program Development, *IBM Systems Journal*, vol. 15, No. 3, pp 182-211, 1976.
6. Gilb, T., *Principles of Software Engineering Management*, Addison-Wesley, 1988.
7. Greer, D. and Bustard, D.W., SERUM - Software Engineering Risk: Understanding and Management, *Int. J. Project & Business Risk*, 1, 4, 373-388, 1997.
8. Greer, D. and Bustard, D.W., Towards an Evolutionary Delivery Strategy based on Risk Analysis, *Proceedings of Engineering of Computer Based Systems*, IEEE Computer Society Press, March, 1996.
9. Greer, D., *Software Risk Analysis and Management Project at NEC Corp.*, University of Ulster, July, 1998.
10. Karlsson, K. and Ryan, K., Prioritizing requirements using a cost-value approach, *IEEE Software*, 14 (5), 67-74, 1997.
11. Karlsson, J, Wohlin, C. and Regnell, B., An evaluation of Methods for Prioritizing Software Requirements, *J Information and Software Technology*, 39, 939-947, 1998
12. Lehman, M.M. and Belady, L.. *Program Evolution: Processes of Software Change*. Acad. Press, London, 1985.
13. Saaty, T.L., *The Analytic Hierarchy Process*, McGraw Hill, 1980.
14. SEI (Software Engineering Institute, Carnegie-Mellon Univesity, USA*), Draft C of SW-CMM v2.0*, 1998.
15. Wilson, B., Systems: Concepts, Methodologies and Applications, 2nd Edition, Wiley, 1990.