Risk Identification Checklist

Risk identification checklists can be helpful when identifying potential risk areas. The following table from Barry Boehm's "Software Risk Management: Principles and Practices" is a listing of the top 10 risks likely to result in a software development project's failure. Included with each risk is a collection of questions that may be asked to simulate discussion, and probe for and identify potential project risks.

1. **Personnel shortfalls**
    a. Are the required staff/skills available for the project time frame?
    b. Do the people have the right combination of skills?
    c. Are enough people available?
    d. Is the staff committed for the entire duration of the project?
    e. Will some project staff members be working only part time on this project?
    f. Does the staff have the right expectations about the job at hand?
    g. Have the staff members received necessary training or is training planned and scheduled?
    h. Will turnover among staff members be low enough to allow continuity?
    i. Does the estimated staffing profile seem reasonable given current organization resource availability?
2. **Unrealistic schedules and budgets**
    a. Were all product components estimated?
    b. Were all tasks performed during standard development included in the schedule?
    c. Were all supporting tasks for the development effort estimated?
    d. Were all affected groups involved in their estimates and have they agreed and committed to them?
    e. Does the work seem doable in the schedule and effort bid?
    f. Is the man month (hours per month) bid reasonable and for the organization?
    g. Has adequate ramp up time been provided for staffing the effort and for the staff to come up to speed on the project?
    h. Is the budgeted cost schedule possible given current resource availability?
    i. Have all dependencies between tasks been identified and included in schedule?
    j. Have all dependencies between affected project groups been identified (horizontal schedule integration) and included in the schedule?
3. **Developing the wrong functions and properties**
    a. Are the requirements for the system documented?
    b. Are the requirements for the system clear and understandable?
    c. Is there a history of misinterpretation or miscommunication between developers and those defining product requirements?
    d. Are the requirements stable?
    e. Is there a history of volatility in product requirements?
    f. Do requirements have a history of changing after product release to test?

4. **Developing the wrong user interface**
    a. Has the user interface been prototyped and tested on sample user groups?
    b. Have surveys on user needs and expectations been performed?
    c. Have user mental models been examined and analyzed in relationship to the user interface?
    d. Are user interface goals defined and documented?
    e. Have user work scenarios been defined and analyzed in terms of interface goals?
5. **Gold plating**
    a. Is there a history of unneeded functionality being added during project development?
6. **Continuing stream of requirements changes**
    a. Are the requirements for the project well defined and baselined?
    b. Are there known change requests that are pending?
    c. Has past history with customer shown that requirements change often?
    d. Are there existing products on the market with similar functionality that can be used to help baseline or establish firm requirements?
    e. Have market surveys defining desired user functionality been performed?
    f. Does the team have a good understanding of what the customer/user wants in the system?
7. **Shortfalls in externally furnished components**
    a. Are there any delivery risks identified with projects to which this one is dependent?
    b. Are there any technology or functionality risks identified with the projects to which this one is dependent?
8. **Shortfalls in externally performed tasks**
    a. Is this project or any of its tasks dependent upon the completion of external project development efforts?
    b. Are layered products necessary to support functionality readily available?
9. **Real-time performance shortfalls**
    a. Have requirements for system response to user functions been defined?
    b. Have requirements for data storage been defined?
    c. Have requirements for memory usage been defined
    d. Have processing speeds for system performance been defined?
    e. Have message processing or data throughput requirements been defined?
    f. Have user support (number of) requirements been defined?
10. **Straining computer-science capabilities**
    a. Has the system or type of system been accomplished before by the organization?
    b. Can current processing capabilities support performance requirements?
    c. Is the computer technology necessary to support required functionality readily available?
    d. Are the programming languages familiar and known to staff?
    e. Is the computer architecture (OS, layered products, etc.) familiar and known to the staff?